A D A L A B (tm) H A R D W A R E M A N U A L

By Paul K. Warme

INTERACTIVE MICROWARE, INC.

AN OVERVIEW OF THE ADALAB(tm) DATA ACQUISITION SYSTEM

ADALAB is a microcomputer system that is specifically designed for APPlications in the LABoratory. We call ADALAB a system because it includes both the hardware (the APPLE computer and ADALAB interface board) and extensive software to make this computer useful in every lab. Any instrument that can output voltages to a recorder may be connected to ADALAB. Some examples of such instruments are spectrophotometers, fluorometers, flame photometers, pH meters, chromatography monitors, gas chromatographs, HPLC systems, conductivity meters and oxygen monitors. Instruments that are controlled by a voltage may also be connected to ADALAB. This category includes strip chart recorders, proportional control valves and pumps, temperature or flow controllers and electrochemical instruments. ADALAB is also useful for controlling any instrument that has multiple switch inputs or contact closure outputs. In addition, ADALAB can communicate with "intelligent" instruments or with other computers having parallel or serial (optional) input/output capabilities.

ADALAB HARDWARE SUMMARY

>>> Analog to Digital (A/D) Converter Subsystem
* Reads voltages from your instruments with a precision of 0.025% (12 bits) and overall accuracy adjustable to better than 0.1%
* Jumper-selectable voltage ranges ±4V, ±2V, ±1V and ±0.5V
* Dual slope integrating A/D converter smooths out noisy signals
* True differential input and automatic internal zeroing enhance accuracy
* Up to 20 voltage readings per second; faster response than most recorders

>>> Digital to Analog (D/A) Converter Subsystem
* Sends control voltages to your instruments with 0.025% precision (12 bits) and overall accuracy better than 0.1%
* Jumper-selectable voltage ranges ±4V, ±2V, ±1V and ±0.5V
* D/A conversion rate up to 50,000 conversions per second

>>> Digital and Parallel Input/Output Subsystem
* 8 digital input bits and 8 digital output bits or 16 bidirectional bits individually selectable as inputs or outputs
* TTL-compatible signal levels (one TTL load or drive)
* Versatile handshaking signals, interrupt and enable circuitry
* Latching registers store I/O information on cue

>>> Real-Time Clock/Timer Subsystem
* 32 bit countdown timer may be set for any time interval from 10 microseconds to 100 minutes. May be programmed as a time of day clock reading in hours, minutes and seconds
* Two 16 bit timer/counters may be configured as an interval timer, event counter, pulse generator, square wave generator or shift register

INSTALLING YOUR ADALAB(tm) INTERFACE CARD

Unpacking

The following items are included with each ADALAB(tm) interface card:

3	16 pin DIP cables (36")
1	Self-Test Adapter Module
1	ADALAB Hardware Manual
1	QUICKI/O Software Manual
1	QUICKI/O Software Diskette
1	Warranty Card and Registration Form
1	Evaluation Form

STATIC WARNING: The large (24 and 40 pin) integrated circuits on the ADALAB card may be damaged by static electricity. Before removing the protective wrapping or handling the ADALAB card, you should ground yourself by touching a water faucet or the metal case on the APPLE computer's power supply.

Selecting Jumper Options

Looking at the ADALAB card on the component side with the gold edge connector at the bottom right, you will see three groups of metal pins on the upper half of the card (see Figure 1). Group 1 is a vertical column of 2 by 8 pins near the center of the board; Group 2 is a vertical column of 2 by 4 pins, and Group 3 is a horizontal row of 2 by 7 pins. On some of these pins, you will see black plastic jumpers, which connect pairs of pins together. As shipped, the jumpers are set for the ±4V range and slot 2; thus, if you plug the ADALAB card into slot 2 of your APPLE computer, no jumper changes are required.

GROUP 1 Jumper Options (A/D Converter)

The two jumpers on the Group 1 pins are used to select the range of the Analog to Digital (A/D) converter. For the ±4 Volt range, one jumper should be on the pair of pins closest to the top edge of the card and the other jumper should be on the fifth pair down from the top edge. For the ±2V range, move each jumper down one position. If you move each jumper down one more position, you will select the ±1V range. Likewise, moving down one more position selects the ±0.5V range for the A/D converter. Note that there should always be three vacant pairs of pins between the two jumpers on the Group 1 pins. It will help if you remember that the voltage range decreases as you move the jumpers downward.

GROUP 2 Jumper Options (D/A Converter)

The second group of pins, running vertically on the upper right (see Figure 1) selects the voltage range for the D/A converter. As was the case for GROUP 1, the voltage range decreases from $\pm$4V (top pair) to $\pm$0.5V (bottom pair) as you move the single jumper downward.

GROUP 3 Jumper Options (Slot Selection)

You may insert the ADALAB interface card into any one of slots 1 through 7 in your APPLE computer. Note that slot 0 is reserved for language cards and therefore, the ADALAB card must not be used in slot 0. The seven pairs of pins that run horizontally near the top right corner (see Figure 1) select the slot number. To select slot 1, place the jumper on the first pair of pins, counting from the left. Move the jumper one position to the right for slot 2, one more position to select slot 3, and so on. If the jumper is not placed on the pins that correspond to the slot you are using for the ADALAB card, the software will not be able to communicate data properly.

Cable Attachment

Looking at the ADALAB interface card or referring to Figure 1, you will note that there are three empty 16-pin sockets which are used for attaching the ribbon cables. The Digital (or Parallel) Output socket is at the upper left corner, the Digital Input socket is directly to the right of it, and the Analog Input/Output socket is at the upper right corner of the ADALAB card. The signals on each pin of these sockets are summarized in Table I. Pin 1 is in the upper right corner of each socket. It is IMPORTANT to connect the cables in such a way that pin 1 of the cable plug (the pin marked with a black dot or triangle) is inserted in the upper right corner of the socket (the corner that is bevelled at a 45 degree angle). Since the plugs on the two ends of the cable are opposite in orientation, it is possible to plug in the cables so that they extend either upward or downward. Just be sure that the black triangle is in the upper right corner of the socket. Now, remove the cover on your APPLE computer, insert the ADALAB card in the slot selected and run the cables out through one of the notches in the back of the computer. If the cables extend upward, fold them over the top edge of the board and run the cables along the back of the board. This will avoid interference when you put the cover of the APPLE computer back on.

Connecting Cables to Adapter Modules

When you connect the ADALAB cables to the self-test adapter or other signal conditioner modules, be sure that the black dot or triangle near pin 1 is inserted in pin 1 of the socket, labelled with a dot or triangle. Also, be sure to connect each cable only to the proper socket.

WARNING: The analog I/O cable should only be plugged into a socket marked ANALOG. The digital input cable should only be plugged into a socket marked INPUT or DIGITAL INPUT, and the digital output cable should go only in a socket marked OUTPUT or DIGITAL OUTPUT. Improper insertion of the cables could cause permanent damage!

CALIBRATION PROCEDURES

The ADALAB interface card is calibrated at the factory for +4V operation. If this is satisfactory, you may proceed to the QUICKI/O Software Manual and try the QUICKSAMPLE demonstration program.

After changing the voltage range jumper options, as described in the previous section, you should recalibrate the D/A and A/D converters. To do this, you should use a high-quality digital voltmeter with at least four digit accuracy. A less accurate voltmeter may be used, but the overall accuracy of the calibration will only be as accurate as your voltmeter.

First, plug the three cables into the self-test adapter. Then, insert the QUICKI/O disk and turn on the computer or, if it is already on, type RUN QUICKSAMPLE.

The QUICKSAMPLE program determines which slot the ADALAB card is in, initializes the hardware and then runs the self-test routines on Channel Ø. You may consult the QUICKI/O Software Manual for additional information about the QUICKSAMPLE program and the self-test procedure. For our present purposes, it is sufficient to know that the self-test procedure ends with the analog I/O test, which allows you to adjust the output voltage of the D/A converter by repeatedly pressing the left or right arrow keys. In order to calibrate the D/A voltage range, press the right arrow key repeatedly until VOUT=2Ø47 (this will be printed near the left side of the screen). If you are using a voltmeter that also measures resistance and current, set it up to measure voltage and use a voltage range of 4 volts or more. Attach the + lead of your voltmeter to the connector on the self-test adapter marked + and attach the - lead (ground) of your voltmeter to the - lead on the self-test adapter. The voltage reading on your voltmeter should now measure close to the maximum voltage that you have selected by means of the D/A jumper. To adjust the maximum D/A voltage, remove the cover of the Apple computer and use a small screwdriver to turn the screw on the square plastic potentiometer closest to the back of the computer. Turning the screw clockwise decreases the voltage.

After you have adjusted the D/A converter voltage, you should adjust the range of the A/D converter. This is done by turning the screw on the other potentiometer which is closest to the front of the computer. Each time you press the right arrow key, a new value is read by the A/D converter, so you should repeatedly press the right arrow key as you turn the screw on the potentiometer. The A/D converter reading (VIN=) is printed on the screen, and you should turn the screw until VIN=2047. Turning the screw clockwise decreases the voltage. If VIN=4095, this means that the input voltage is beyond the maximum, so you should turn the screw back a little (clockwise). You will note that a small change in the screw position changes the voltage reading, so you must turn the screw in very small increments as you approach the final value.

The A/D converter reading is affected by temperature changes and the reading tends to decrease as the computer warms up. Therefore, it is a good idea to let the computer run for at least 15 minutes before calibrating the A/D converter. For most applications, the actual value returned by the A/D converter is not as important as its linearity, its short-term stability, and its ability to measure <u>changes</u> in voltage precisely.

THE ANALOG TO DIGITAL CONVERTER

Theory of Operation

An A/D conversion sequence begins when a number is stored in the high byte of the digital to analog (D/A) converter output register. However, this has no affect on the D/A converter voltage, because the D/A converter is triggered by storing a number in the low byte of its output register. When the A/D converter finishes the conversion process, it sets a flag which can be read by your program. Thus, your program can readily determine whether the A/D value is ready. Also, it is possible to set up the A/D converter so that it will produce an interrupt when the conversion is completed. Further information about these features is given under Programming Considerations.

ADALAB's A/D converter is called a dual-slope (or integrating) A/D converter. This means that the input voltage is measured by charging a capacitor over a precisely determined interval of time. Very little current is drawn from the external equipment because of the high input impedance of the operational amplifiers used on the A/D inputs. The final integrated charge on the capacitor is proportional to the input voltage. Next, the capacitor is discharged by connecting it to a stable reference voltage. The time required to discharge the capacitor is also proportional to the input voltage. Since the discharge time is a number that can be read by the computer, we obtain a number that is proportional to the input voltage. If we plot the charge on the capacitor as a function of time, we will see a straight line with a certain slope as the capacitor charges and another straight line with a different slope as the capacitor discharges. This is the origin of the term "dual-slope."

There is another common type of A/D converter which is called a "successive approximation" A/D converter. It uses a D/A converter and a voltage comparator to measure voltages. First, the most significant bit of the digital value sent to the D/A converter is turned on and the resulting D/A output voltage is compared to the A/D input voltage that is to be evaluated. If the D/A output voltage is larger than the A/D input voltage, the most significant bit of the digital value sent to the D/A converter is turned off again. Then, the second significant bit of the D/A digital value is turned on and the D/A output voltage is once more compared to the A/D input voltage. In this case, perhaps the D/A voltage is less than the A/D voltage, so that bit of the D/A digital value is left on. This process of turning bits on or off and comparing voltages continues on until each bit of the D/A digital value has been tested by this successive approximation technique. At this point, the D/A output voltage should be exactly the same as the A/D input voltage; therefore, the digital value sent to the D/A converter is an accurate measure of the A/D input voltage.

Both the dual-slope (DS) method and the successive

approximation (SA) method have certain advantages and disadvantages. The successive approximation method is fast; typically, a measurement is completed within 2Ø microseconds. The dual-slope method is considerably slower; typically a conversion takes about 5Ø milliseconds.

The designers of ADALAB chose a dual-slope converter because the successive approximation method runs into problems when the input signal is "noisy", as is the case with the signals created by most laboratory instruments. Consider what would happen if a voltage fluctuation occurred during the period when a SA converter is trying to measure a voltage. Suppose that the voltage is momentarily higher than average. Then one or more bits of the D/A output value will be left on which should have been turned off. As the subsequent bits are tested, the voltage may go back down to the average value, so all subsequent bits will be turned off, because the D/A output voltage is already higher than the A/D input voltage. To counteract this problem, it is usually necessary to use a sample and hold amplifier with a SA converter. The sample and hold amplifier samples the input voltage for a very brief period (aperture time less than one microsecond) and then holds that voltage while the SA converter measures it. But now, the measured voltage contains information only about the instantaneous voltage during that very short aperture time of the sample and hold amplifier. If the fluctuations of the input voltage are significantly large, it will be necessary to average many SA conversion values to obtain an accurate indication of the average input voltage. Because many conversions must be averaged, the effective rate of a SA converter for noisy signals is much slower than the specified maximum rate. Also, extra memory space is needed to store the values to be averaged and the program is more complicated because of the need to average multiple values.

How does the dual-slope method escape this problem? A DS converter automatically averages (integrates) the input signal during a major portion of the conversion time (one-fourth of the time in the case of ADALAB). Thus, fluctuations of the input voltage will affect the rate of charging of the capacitor, but positive fluctuations will be cancelled by negative fluctuations and the resulting value is a true average, equivalent to the average of thousands of measurements using a SA converter with a sample and hold amplifier. As you can see, the DS converter "damps out" voltage fluctuations, similar to the way that a strip-chart recorder damps out noise. Since many laboratory instruments are designed to be connected to strip chart recorders, their output circuitry doesn't attempt to filter out noise that will not normally show up because the recorder is heavily damped. Many scientists are surprised to learn how much noise occurs in the signals they are measuring. To observe this noise, you should attach an oscilloscope to the recorder output of your instrument.

Another feature of ADALAB's A/D converter that contributes

to accuracy is that it automatically zeroes itself between each measurement. This compensates for internal offset errors generated by the buffer amplifier, integrator and comparator. The ADALAB A/D converter also has true differential inputs. In other words, it measures the voltage difference between the high (+) input and the low (-) input. Normally, the low input is close to ground potential, but other electrical equipment in the vicinity can induce voltages in the wires connecting your instrument to ADALAB. Electrical noise is prevalent in most labs, and noise can affect the accuracy of the input voltage, especially when the wires connecting your instrument to the computer are quite long. Normally, induced voltages in the signal lines will affect both the high and low inputs nearly equally, so ADALAB's differential inputs tend to counteract the effects of induced noise.

The ADALAB A/D converter chip is actually a 13 bit A/D converter, but the QUICKI/O software throws away the least significant bit, in order to make the A/D converter readings directly comparable to the 12-bit D/A converter output values. Throwing away the least significant bit also increases the stability of the A/D values, because noise affects primarily the low bits of the value. When comparing the specifications of ADALAB with other A/D converters on the market, you should be aware that most other products available for the APPLE computer are only 8 bit A/D converters, they allow only positive voltage inputs, and they have only a single voltage range. An 8 bit converter returns values in the range 0 to 255, whereas APPLAB's 12 bit converter returns values from -2047 to 2047, using any of 4 different voltage scales. Thus, ADALAB is 16 times more accurate than 8 bit A/D converters and gives you the added advantages of measuring both positive and negative voltages. In addition, jumper-selectable voltage scales enable you to use ADALAB with a wide variety of instruments having different full scale voltage ranges.

A/D Converter Specifications

Integrated Circuit: Intersil 7109 dual-slope A/D converter

Resolution: 12 bits plus sign bit and over-range bit

Full Scale Voltage: ±0.5V, ±1.0V, ±2.0V, or ±4.0V, jumper selectable

Maximum Conversion Time: 50 milliseconds

Minimum Conversion Rate: 20 samples per second

True Differential Input (dual floating inputs)

Autozeroing compensates for internal offset voltages

Maximum Input Voltage: ±12V without damage

Input Impedance: minimum 8 megohms

Input Current: maximum Ø.5 microamperes

Differential Nonlinearity (maximum deviation from ideal step size): ±2 counts (Ø.Ø5%)

Integral Nonlinearity (maximum deviation from ideal straight line): ±4 counts (Ø.1Ø%)

Overall Accuracy: Adjustable to better than Ø.1% of full scale range

Common Mode Rejection Ratio (common mode voltage ±1V, input voltage ØV, full scale voltage ±Ø.5V): 5Ø microvolts per volt

Temperature Coefficient: 1ØØ ppm/degree C

Software Interface: via Initiate Conversion command, Conversion Completed signal, and Interrupt Enable register. Data are read as two 8-bit bytes. The first byte includes the sign and over-range indicators and the most significant 4 bits. The second byte includes the least significant 8 bits of data.

A/D Converter Programming Considerations

This section discusses programming of the A/D converter at the assembler language level. For most applications, you will find it much easier to use the QUICKI/O approach described in the software manual. However, if you wish to use interrupts or polled sampling of the Conversion Completed signal, you should study this section. Here, you will also learn how to obtain the full 13-bit precision that is permitted by the Intersil 71Ø9 chip. All addresses in this section are given in hexadecimal notation.

The A/D converter is controlled by the "dedicated" 6522 chip on the ADALAB interface card. As we shall see later, this "dedicated" 6522 is also used for the real-time clock and the D/A converter. We will call the other 6522 chip the "user" 6522 because its functions are completely under the control of the user. Each of these two 6522 chips has 16 successive memory addresses which control its functions. The addresses for the dedicated 6522 start at address BASE1= $CØØØ + N*$1ØØ, where N is the slot number of the ADALAB card. The addresses for the user 6522 begin at address BASE2=$CØ3Ø + N*$1ØØ. Table III tells the function of each address in the set of 16 addresses associated with the dedicated 6522 chip.

To initialize the dedicated 6522 chip, use this program segment:

```
        LDA #$8F           ;high byte data direction
        STA BASE1+$02
        LDA #$FF           ;low byte data direction
        STA BASE1+$03
        LDA #$BE           ;low byte of timer 0
        STA BASE1+$04
        LDA #$C7           ;high byte of timer 0
        STA BASE1+$05
        LDA #$E0           ;auxiliary control register
        STA BASE1+$0B
        LDA #$8A           ;peripheral control register
        STA BASE1+$0C
```

This initialization program sets up the D/A converter and timers, as well as the A/D converter. Note that the BASE1 address is calculated as described in the previous paragraph.

To start an A/D conversion, you must write any value into address BASE1. As noted in Table III, this is the same as the address of the D/A high byte. However, the D/A high byte does not take effect until the D/A low byte is written into address BASE1+1. Thus, starting the A/D converter doesn't interfere with operation of the D/A converter.

To find out whether the A/D conversion is completed, you must test bit 4 at address BASE1+$0D. This bit is low (0) during an A/D conversion and goes high (1) after completion. The following program will start the A/D and wait for the conversion done signal:

```
        STA BASE1          ;start A/D
WAIT    LDA BASE1+$0D      ;read interrupt flags
        AND#$10            ;check bit 4
        BEQ WAIT           ;loop if not done
DONE    (continue)
```

To enable the A/D converter to interrupt your program after conversion is done, you should set bit 4 of the interrupt enable register. Storing $90 at address BASE1+$0E enables interrupts, while storing $10 disables interrupts. Of course, you must provide an interrupt handler that catches the interrupts and services the A/D converter. If it was the A/D converter that caused the interrupt, bit 4 of BASE1+$0D should be on (1). This subroutine sets up for interrupts by the A/D converter and allows for interrupts by other devices:

```
SETUP   SEI                ;disable IRQ interrupts
        LDA #<IRQINT       ;low byte address of IRQ handler
```

```
        STA $Ø3FE        ;IRQ jump vector for Apple
        LDA #>IRQINT     ;high byte address
        STA $Ø3FF        ;high byte of IRQ jump vector
        CLI              ;reenable interrupts
        RTS

IRQINT  LDA BASE1+ØD     ;is it an A/D interrupt?
        AND #$1Ø
        BEQ OTHER        ;if bit 4=Ø, then not A/D
        AND BASE1+$ØE    ;are A/D interrupts enabled?
        BNE A/DINT       ;yes, if bit 4 is 1
OTHER                    ;other interrupts handled here
A/DINT  LDA BASE1+$1Ø    ;read A/D low byte
        STA ADVALUE      ;store in memory
        LDA BASE1+$2Ø    ;A/D high byte
        STA ADVALUE+$Ø1  ;store it
        STA BASE1        ;start A/D converter again
        LDA $45          ;recover A register value from before
                         interrupt
        RTI              ;return from interrupt
ADVALUE ØØØØ             ;two bytes to store A/D value
```

The above routine will continuously run the A/D converter at maximum rate. It stores the most recent value in ADVALUE.

Your program should read the A/D converter low byte value at address BASE1+$1Ø and the high byte at BASE1+$2Ø. The high byte value contains the four most significant bits of the answer (in bits Ø-3), the overrange indicator (bit 4) and the sign bit (bit 5). Bits 6 and 7 of the high byte are unused and unspecified, but generally they read as 1's (on). Table II shows the binary and hexadecimal codes that correspond to various input voltages. The following subroutine converts the raw A/D value in ADVALUE into a ones complement number ranging from -8192 to 8191 (decimal) or $EØØØ to $1FFF (hex).

```
CONVERT   LDA ADVALUE+$Ø1    ;high byte
          AND #$2Ø           ;check for negative value
          BNE PLUS           ;bit 5 on means plus
MINUS     LDA ADVALUE        ;complement low byte if minus
          EOR #$FF
          STA ADVALUE
          LDA ADVALUE+$Ø1    ;complement high byte
          EOR #$FF
          ORA #$EØ           ;turn on all sign bits
          STA ADVALUE+$Ø1
          RTS
PLUS      LDA ADVALUE+$Ø1    ;load high byte
          AND #$1F           ;zero out high bits
          STA ADVALUE+$Ø1
          RTS
```

A/D CONNECTIONS AND INTERFACING

Table I lists the pin assignments on the analog I/O socket and cable. The analog I/O socket is in the upper right corner of the ADALAB card, as shown in Fig. 1. Normally, you should connect the ground wire of your instruments to pin 12 (A/D low) and connect the varying voltage signal to pin 1Ø (A/D high). Since the A/D converter can measure both positive and negative voltages with equal ease, you will not cause any damage if you reverse the wires.

WARNING: Do not connect any device which may exceed +5V or -5V because permanent damage may result.

THE DIGITAL TO ANALOG CONVERTER

Theory of Operation

The ADALAB D/A converter has 12 bit precision, but it is set up so that you can output the data in two separate 8-bit data units. First, the most significant 4 bits of data are stored in a particular memory location, but this does not change the D/A output voltage (yet). When the remaining 8 bits of data are stored in the next subsequent location, the most significant 4 bits of data are transferred (latched) into an output register. Thus, all 12 bits of the new data are presented simultaneously to the D/A converter. The response time of the D/A converter is almost instantaneous (settling time for a full scale voltage change is only about 3 microseconds). Thus, no status bit or interrupt is needed to tell us when the conversion is completed. However, the speed of the D/A converter is limited by the software because as we shall soon see, a simple program loop to output 12 bits of data takes about 20 microseconds. It is possible to operate the D/A converter at a faster rate if you don't change the high 4 bits and only update the low 8 bits.

How does the D/A converter produce an analog output voltage when given a particular digital input value? As shown in Fig. 2, the D/A circuit consists of a stable reference voltage (VREF), a set of switches (one for each digital bit), a set of carefully matched resistors and a summing operational amplifier (opamp). Each digital input bit controls one of the switches. Any switch that is connected to VREF will cause current to flow through the resistor network (sometimes called a "ladder") to the summing junction of the opamp. The clever thing about this circuit is that the output voltage is the binary weighted sum of the digital input bits, multiplied by the reference voltage. In other words, the output analog voltage is proportional to the digital input value.

Although the D/A converter chip is programmed for ±5V operation, its output voltage is divided down by a chain of resistors. This provides jumper-selectable ranges of ±4V, ±2V, ±1V and ±0.5V. The output from this voltage divider is buffered by an operational amplifier operating as a voltage follower with a gain of one. Thus, the D/A output voltage has considerable current (low output impedance) to drive external equipment.

D/A Converter Specifications

Integrated Circuit: Analog Devices DAC80

Resolution: 12 bits

Full Scale Voltage: ±0.5V, ±1.0V, ±2.0V or ±4.0V, jumper selectable

Maximum Conversion Time: 20 microseconds

Minimum Conversion Rate: up to 50,000 conversions per second, limited only by software speed.

Output Current: sources or sinks 10mA

Nonlinearity: ±1 least significant bit

Monotonic: over entire 0 to 70 degree C range

Accuracy: Adjustable to better than 0.1% of full scale range

Temperature Coefficient: 100 ppm/degree C

Software Interface: via output of two data bytes; the most significant 4 bits are stored until the least significant 8 bits are output and then the 12 bits of data are presented simultaneously to the D/A converter.

D/A Converter Programming Considerations

The easiest way to use the D/A converter is to program it with QUICKI/O, as described in the software manual. However, this section explains how to program the D/A converter in assembler language, which enables the D/A converter to run at rates of up to 50,000 conversions per second.

The D/A converter is controlled by the dedicated 6522 chip. Thus, to initialize the D/A, you must store $0F in location BASE1+$02, store $FF in location BASE1+$03 and store $8A in location BASE1+$0C. This sets up the 6522 chip for output of 12 bits, with triggering of the high byte latch as soon as the low byte is stored. A program to initialize the dedicated 6522 chip was presented earlier (see A/D Converter Programming Considerations).

To output a voltage on the D/A converter, first store the most significant 4 bits in location BASE1 and then store the least significant 8 bits in location BASE1+$01. That's all there is to it. It doesn't matter what you place in the high order 4 bits of location BASE1 because only the low 4 bits are actually used as the most significant 4 bits of the output voltage.

The D/A converter uses a complementary offset binary format. Table II shows the digital codes for various output voltages. As

you can see, QUICKI/O has to perform some mathematical manipulations in order to make digital codes -2047 to 2047 correspond to minus full scale through plus full scale voltage. The easiest way to transform a signed 16 bit binary value from -2047 ($F801) to 2047 ($07FF) into the appropriate form for the D/A converter is as follows:

```
        LDA DALOW       ;D/A low byte
        EOR #$FF        ;two's complement
        CLC
        ADC #$01
        PHA             ;stack for output
        LDA DAHIGH      ;D/A high byte
        EOR #$F7        ;two's complement; reverse bit 3
        ADC #$00        ;add carry from low byte
        STA BASE1       ;slot dependent address
        PLA             ;unstack low byte
        STA BASE1+01    ;low byte triggers high latch
```

Bear in mind that each time you output a value to BASE1, it automatically triggers the A/D converter. If the A/D converter is in the middle of a conversion, an extra trigger will have no effect on it.

If you are interested in running the D/A converter at a very fast rate, here are some programming tips. In all of these cases, we will use the X (or Y) register to index an array of data values. Faster rates are attainable if the data are stored in page 0, but usually this is not feasible. First, let's convert samples from a table of 256 8-bit values, holding the high byte constant (12 machine cycles per loop or 83.3 KHz rate for 1MHz clock).

```
      LDA DAHIGH        ;D/A high byte
      STA BASE1
      LDX #$00          ;point to first data byte
DAOUT LDA DATADR,X
      STA BASE1+01
      INX               ;increment pointer
      BNE DAOUT         ;loop for 256 values
      BEQ DAOUT         ;include this for continuous output
```

To refresh both the high byte and the low byte at maximum rate, store the high byte and low byte data as two separate tables and use this program (20 clock periods per loop or 50 KHz output rate):

```
      LDX #$00          ;point to first data byte
DAOUT LDA DATAHI,X      ;high byte
      STA BASE1
      LDA DATALO,X      ;low byte
```

```
STA BASE1+$01
INX              ;increment pinter
BNE DAOUT
BEQ DAOUT        ;include this for continuous output
```

For each of the previous two programs, the BEQ DAOUT instruction at the end makes a continuous waveform output. To change the frequency of the output waveform, we can advance X by a SKIP interval different from one. Replace the INX instruction above with the follwing code (this adds 4 to 6 clock periods per loop, depending on the addressing mode of the ADC command):

```
TXA        ;current position in data
ADC SKIP   ;change SKIP for different frequency
TAX        ;ignore overflow
```

D/A Connections and Interfacing

As indicated in Table I, the D/A output voltage (high) is on pin 13 of the analog I/O socket, which is in the upper right corner of the ADALAB interface card (see Fig. 1). The D/A reference voltage (low) is the analog ground on pin 11. Note that the D/A output voltage may be either positive or negative, relative to analog ground.

WARNING: When the computer is first turned on, the output voltage is the most <u>negative</u> voltage for the range selected by the D/A jumper. If <u>you are</u> connecting the D/A converter to an instrument that cannot tolerate negative voltages, be sure to disconnect the cable before turning the computer on.

DIGITAL (PARALLEL) INPUT AND OUTPUT

Theory of Operation

You will recall that the QUICKI/O software distinguishes between digital (bitwise) I/O and parallel (bytewise) I/O. At the hardware level, there is no difference between digital and parallel I/O. The ADALAB parallel I/O is implemented as part of the user 6522 chip, which is totally moldable to your desires. Since there are no parallel I/O buffers on the ADALAB card, each of the 16 parallel I/O bits may be selected for either input or output. Table IV lists the addresses that control each function of the user 6522 chip. In the data direction registers for port A and port B, each bit that is on (1) is selected for output, while each bit that is off (Ø) is selected for input. Normally, port B is used for output because it has greater current drive capability than port A. When timer 2 is used as a frequency generator, bit 7 of port B must be selected for output. When timer 3 is used as a pulse counter, bit 6 of port B must be selected for input.

Four handshaking lines are available to facilitate and synchronize communications between devices. The CA1 and CB1 lines may be set up to recognize either positive or negative input transitions and each can set a bit in the interrupt flag register when such a transition occurs. If the corresponding bit in the interrupt enable register is set, a transition on CA1 or CB1 will generate an interrupt. In addition, transitions on CA1 or CB1 will cause latching of the input or output data if the Auxiliary Control Register is set up appropriately.

Handshaking lines CA2 and CB2 have the same input capabilities as lines CA1 and CB1, but they also can output signals in four different modes. In mode 1, CA2 and CB2 will change state when data is read from or written into port A or port B, respectively. Their state reverses again when an active transition occurs on CA1 or CB1, respectively. This is exactly what we need for automatic handshaking. In mode 2, a short pulse (one microsecond) is sent out on CA2 or CB2 when data is read from or written to port A or port B. In mode 3, CA2 and CB2 are held low, whereas these outputs are held high in mode 4.

Digital I/O Specifications*

Integrated Circuit: MOS Technology 6522 Versatile Interface Adapter

16 bidirectional lines (usually used as 8 bits in and 8 bits out)

Latching capability on input or output

4 handshaking signals accommodate positive or negative logic

Interrupt register and interrupt enable register for each handshake signal.

Input Characteristics:
- High Voltage: 2.4V to 5.ØV
  - Current: -1ØØ to -25Ø microamperes
- Low Voltage: -Ø.3V to +Ø.4V
  - Current: -1.Ø to -1.6 milliamperes
- Leakage Current: ±1.Ø to ±2.5 microamperes
- Off-state Current: ±2.Ø to ±1Ø microamperes
- Capacitance: 1Ø pF

Output Characteristics:
- High Voltage: 2.4V minimum
  - Current: -Ø.1 to -1.Ø milliamperes (PAØ-PA7, CA2)
    -3.Ø to -5.Ø milliamperes (PBØ-PB7, CB1, CB2)
- Low Voltage: Ø.4V maximum
  - Current: 1.6 milliamperes
- Leakage Current: 1.Ø-1Ø microamperes
- Capacitance: 1Ø pF

* See also the Versatile Interface Adapter Data Sheets

Digital I/O Programming Considerations

If QUICKI/O fails to meet your requirements, this part of the manual will tell you how to program the digital I/O using assembler language. The assembler language approach is necessary if you want to use some combination other than 8 bits in on Port A and 8 bits out on Port B.

The direction of data flow is controlled by writing a one (1) for each output bit or a zero (Ø) for each input bit into address BASE2+2 (port B) or address BASE2+3 (port A). After setting up the data direction, you can read or write data to address BASE2 (port B) or BASE2+1 (port A). In general, you can read or write to a parallel port, regardless of whether it is selected for input or output. If you read a bit selected for output, you will obtain the last value stored for that bit. If you write to a bit selected for input, it will have no effect. As indicated in Table V, you may enable latching of data for Ports A and/or B by setting the proper bits in the auxiliary control register. When latching is selected, a handshaking transition on CA1 or CB1 will cause the current data to be latched (stored) until it is needed. This feature allows ADALAB

to capture momentary data on cue from some external device. If latching is not enabled, the values read from a port will reflect the current input data.

The Peripheral Control Register (see Table VI) is very important for selecting the type of handshaking to be used for digital I/O. The low order 4 bits control the handshaking for Port A, while the high order 4 bits pertain to Port B. CA1 and CB1 are always input lines, capable of detecting either positive or negative transitions produced by your external equipment. CA2 and CB2 are more versatile; they can be used as either inputs or outputs. In the input modes, you have a choice of either positive or negative transitions, as well as two different ways of clearing the interrupt flag. In the output modes, you have a choice of constant voltage level outputs (handshake or manual modes) or pulse outputs. The handshake mode and pulse mode are particularly convenient to use, because they coordinate the operation of the CA1+CA2 or CB1+CB2 handshaking pairs.

For example, let us set up Port A as an input and Port B as an output, with both ports using the handshake mode of operation. We will assume that the Port B data lines are connected directly to the Port A input data lines and that CA1 is connected to CB2, while CB1 is connected to CA2. In other words, this is exactly the way the self-test adapter is connected; it is also the way most instruments operate when using digital I/O.

```
LDA#$00             ;Port A data direction
STA BASE2+$03
LDA #$FF            ;Port B data direction
STA BASE2+$02
LDA BASE2+$0B       ;Read auxiliary control register
AND #$FC            ;Preserve bits 2 to 7
ORA #$01            ;Enable port A latch
STA BASE2+$0B       ;Save auxiliary control register
LDA #$88            ;Handshake on ports A and B
STA BASE2+$0C       ;Peripheral Control Register
```

Now, everything is initialized. Let's make a dry run to see how the handshaking works. First, we write data to Port B. This makes CB2 go low and, since CB2 is connected to CA1, CA1 also goes low. Because CA1 is low, CA2 goes high and the CA1 interrupt flag is set. This ensures that the input program will know that data is ready. Next, the input program reads the data, which causes CA2 to go low again. Since CA2 is connected to CB1, CB2 goes high, which in turn sets the CB1 interrupt flag. This tells the output program that the last data was received and so, it is time to send new data. The following program could be used to output a group of 10 data points stored at address DATAOUT:

```
            LDX #$00            ;set up counter
OUTWAIT     LDA BASE2+$0D       ;read interrupt flag register
            AND #$10            ;isolate bit 4
            BEQ OUTWAIT         ;off means not ready
```

```
          LDA DATAOUT,X     ;get data from memory
          STA BASE2         ;output to Port B
          INX               ;increment counter
          CPX #$0A          ;10 values done?
          BMI OUTWAIT       ;loop if not
```

To input 10 values and store them at address DATAIN, this program could be used:

```
          LDX #$00          ;set up counter
INWAIT    LDA BASE2+$0D     ;read interrupt flag register
          AND #$02          ;isolate bit 1
          BEQ IN WAIT       ;off means not ready
          LDA BASE2+$01     ;read input value
          STA DATAIN,X      ;store in memory
          INX               ;increment counter
          CPX #$0A          ;10 values done?
          BMI INWAIT        ;loop if not
```

ADALAB's hardware is capable of generating interrupts when the handshaking lines indicate that it is time to input or output digital data. Table VII lists the bit positions for the interrupt flags associated with various functions of the user 6522. One way to use the interrupt flags is the polling method, as used in the examples above. The other method is the true interrupt approach, whereby the interrupting condition causes the computer to stop what it is doing (the "main" program) and instead, the computer runs a subroutine called an "interrupt handler." The interrupt handler inputs or outputs some data and then returns to the main program. In order to enable interrupts, the appropriate bit in the interrupt enable register must be turned on. For example, to enable interrupts when CA1 is triggered:

```
          LDA #$82          ;bit 7=1 means enable, bit 2 means CA1
          STA BASE2+$0E     ;interrupt enable register
```

To disable interrupts, store $02 in the same place. The following simple interrupt routine inputs a byte of data when a CA1 interrupt occurs:

```
SETUP     LDA #$00          ;initialize pointer
          STA POINTER
          SEI               ;disable interrupts during setup
          LDA #<INTCA1      ;low byte service address
          STA $03FE         ;IRQ jump vector low byte
          LDA #>INTCA1      ;high byte service address
          STA $03FF         ;high byte of IRQ jump vector
          LDA #$82          ;allow interrupts on CA1
          STA BASE2+$0E     interrupt enable register
          CLI               ;enable interrupts
          RTS
```

```
INTCA1      TXA                 ;save X register
            PHA                 ;on stack
            LDX POINTER         ;recover pointer in X
            LDA BASE2+$01       ;read input data
            STA DATAIN,X        ;store in memory
            INX                 ;increment pointer
            STX POINTER
            PLA                 ;unstack X
            TAX
            LDA $45             ;recover A at time of interrupt
            RTI                 ;return from interrupt
```

Digital I/O Connections and Interfacing

Port A and Port B have individual sockets on the APPLAB interface card, as indicated in Fig. 1. The pin assignments are detailed in Table I . Pin 1 on the board is on the top right side of each socket, and the cable should be plugged into the socket with the arrow closest to pin 1.

The current capability of the digital I/O ports is quite limited; they will source or sink only one TTL load. Also, input and output voltages must always be within the range of 0 to 5 volts. If your input or output requirements are different from these conditions, you will need to purchase or build a signal conditioning adapter to bring your signals within these specifications. If you need assistance with this, please call or write Interactive Microware, Inc. for a quote to your specifications.

The Real-Time Clock and Counter/Timers

Theory of Operation

The ADALAB interface card includes four 16 bit timers; two on each of the 6522 Versatile Interface Adapter chips. The two timers on the dedicated 6522 chip (Timers Ø and 1) are ganged together to form a 32 bit timer that is used as the real time clock in QUICKI/O. If you have more than one ADALAB card, you can use the 32 bit timers on the second and subsequent boards as general-purpose timers. The two 16 bit timers located on the user 6522 chip (Timers 2 and 3) are completely available for use as a pulse generator, pulse counter, shift register or square wave generator. Please note that in the following descriptions, the timers are numbered as in QUICKI/O. That is, Timers Ø and 2 correspond to timer 1 in the VIA description sheets, whereas Timers 1 and 3 here correspond to timer 2 in the VIA description. Also, if you have more than one ADALAB card, Timers 4, 8 and 12 are identical to Timer Ø, while Timers 5, 9 and 13 are the same as Timer 1, and so on.

ADALAB's real-time clock is implemented partly in hardware (the dedicated 6522 timers) and partly in software (the timer subroutines in QUICKI/O). Let us briefly consider how this works. During initialization of QUICKI/O, the preset count for timer Ø is set to $C7BE (51,134 decimal), and the auxiliary control register mode is set to $EØ (See Table V). This mode makes timer Ø operate in the free-running mode, generating continuous interrupts at 5Ø millisecond intervals and inverting the state of bit 7 of Port B (PB7) at the same rate. The real-time clock is very accurate because timer Ø is driven by the quartz crystal oscillator (1.Ø23 MHZ) in the APPLE computer. Timer 1 is set up to count pulses on bit 6 of Port B (PB6). Because PB7 is connected to PB6, the value in Timer 1 counts down at the rate of 1Ø counts per second. Since timer 1 is a 16 bit counter, it can count down from 32767 to -32767, a total of 65,535 counts. At 1Ø counts per second, a maximum time interval of 6553.5 seconds or 1.82 hours is possible before Timer 1 finishes counting and wraps around from -32767 back to 32767.

In QUICKI/O, the 5Ø millisecond interrupts from Timer Ø are used to update the hours, minutes, seconds and milliseconds counts. If the seconds count is updated, the display at the upper right of the screen is also updated. The program checks to see whether the "software time" (as displayed on the screen) is the same as the "hardware time" (as measured by the count in Timer 1). If not, the software time is updated at a very fast pace until it catches up with the hardware time. The software time will fall behind when interrupts are disabled, but the hardware time runs constantly, independent of interrupts. Thus, the software is able to detect when interrupts have been disabled and the software time can be corrected when interrupts are reenabled. Bear in mind that interrupts are disabled by pressing

RESET or by executing a SEI instruction. In addition, interrupts are temporarily disabled whenever the disk is reading or writing information.

The two 16 bit timers on the user 6522 are completely available for you to configure as you wish, as summarized in Table V. You may use Timer 2 in the one shot mode or in the free-running mode. In either case, an interrupt flag is set when Timer 2 counts down to Ø, and an interrupt will occur if the interrupt enable register is set up properly. By appropriate initialization of the auxiliary control register, Timer 2 can be made to output a square wave on bit 7 of Port B. This method for producing a square wave signal is particularly convenient because after initialization, it runs automatically, without any further involvement of the microprocessor. The square wave frequency can range from about 8HZ to 167KHZ, with 16 bit resolution between these extremes.

Timer 3 may be used as a very precise interval timer. After an initial count is written to Timer 3, the count is decremented at the processor clock rate (1.Ø23MHZ) and when the count reaches Ø, the interrupt flag is set. If interrupts are enabled, an IRQ interrupt will occur (see Table VII). The count continues to decrement, so it is possible to determine how long it has been since the interrupt flag was set. In the second mode of Timer 3, it counts pulses on bit 6 of Port B. You will recall that we used this feature to create a 32 bit timer on the user 6522 by connecting bits 6 and 7 of Port B together. However, you could use this timer mode to count pulses coming from any external source. When the count reaches zero, the interrupt flag is set and, if the interrupt enable bit is also set, an IRQ interrupt will result. Timers 2 and 3 could also be used together as a frequency counter; Timer 2 could be used to count the time while Timer 3 is used to count a specified number of pulses coming from some external source. To use the frequency counter, read the time from Timer 2 and store a preset count in Timer 3. When an interrupt occurs for Timer 3, read the time from Timer 2 again and calculate the frequency from the ratio of the preset count (Timer 3) to the time elapsed (Timer 2).

The user 6522 chip also has an 8-bit shift register that can input or output serial information. The various modes of this shift register are listed in Table V. The serial data is input or output on handshake line CB2, whereas the shift timing pulses are input or output on handshake line CB1. There are three possible sources of timing pulses: Timer 2, the processor clock (1.Ø23MHZ) or an external clock supplied by your instrument. The shifting operation is initiated by reading from or writing to the shift register. When shifting out, bit 7 is the first bit output and each successive bit is recirculated back into bit Ø. When shifting in, bits initially enter bit Ø and they are shifted towards bit 7. After 8 bits of data have been shifted in or out of the shift register, the interrupt flag is set and an IRQ interrupt will occur if the corresponding interrupt enable bit is

set (see Table VII). In all shift register modes except the free-running mode, the shifting operations stop after 8 bits. In the free-running output mode, the data are continuously shifted out at the Timer 2 rate. This feature could be used to generate complex repeating waveforms that are much more interesting than a simple square wave.

Real Time Clock and Counter/Timer Specifications

Integrated Circuit: Two MOS Technology 6522 Versatile Interface Adapters

Dedicated 6522 has bits 6 and 7 of Port B connected to allow operation of Timers Ø and 1 together as a 32 bit timer for use as a real-time clock.

User 6522 has all functions of Timers 2 and 3 available for user configuration.

Timers Ø and 2: 16 bit countdown timers can be used as:
* one-shot interval timers with optional pulse output on PB7
* continuous frequency generator with optional square wave output on PB7

Timers 1 and 3: 16 bit countdown timers can be used as:
* one-shot interval timers
* frequency counter that counts a predetermined number of pulses on PB6
* shift register rate generator

Shift register: Inputs or outputs 8-bit serial data with timing pulses supplied by Timers 1 or 3, the 1.Ø23MHZ processor clock or an external clock.

Interrupt Control: Interrupt flag and interrupt enable on all functions.

Signal Characteristics: TTL compatible signals (one TTL load or drive)

Real Time Clock and Counter/Timer Programming Considerations

The QUICKI/O software provides the most convenient way to use the real time clock and counter/timers. However, there are some applications that require non-standard use of these features. For example, you might want a real-time clock that ticks faster or slower than 2Ø ticks per second or you might want to use the shift register as a serial I/O port. This section of

the manual provides detailed information about assembly language programming of the various clocks and timers included with ADALAB.

Tables III and IV list the addresses used for access to and control of the dedicated 6522 and the user 6522. Table V explains the function of each bit in the auxiliary control registers and Table VII contains information about the interrupt register and the interrupt enable register. Additional detailed information about the 6522 chip will be found in the VIA description sheets. Now, if all of this seems a bit complicated, you have come to the right conclusion. The 6522 chip has so many features and capabilities that we have to put up with this complexity in order to gain the versatility of its functions.

Timers Ø and 1 can be used as a 32 bit timer because bits 6 and 7 of Port B are connected together. This code will initialize timer Ø to interrupt 2Ø times per second and timer 1 will count down at 1Ø counts per second:

```
SETUP     LDA #$8F          ;Bits Ø-3 and 7 for output, 4-6 for input
          STA BASE1+$Ø2     ;Port B Data Direction Register
          LDA #$BE          ;5Ø milliseconds low byte
          STA BASE1+$Ø4     ;Timer Ø low byte latch
          LDA #$C7          ;5Ø milliseconds high byte
          STA BASE1+$Ø5     ;Timer Ø high byte latch
          LDA #$EØ          ;Timer Ø free-running, Timer 1 counts pulses
          STA BASE1+$ØB     ;Auxiliary Control Register
          SEI               ;Disable interrupts
          LDA #<TIMINT      ;Low address of timer interrupt routine
          STA $Ø3FE         ;IRQ jump vector low byte
          LDA #>TIMINT      ;High address of interrupt routine
          STA $Ø3FF         ;IRQ jump vector high byte
          LDA #$CØ          ;enable Timer Ø interrupts
          STA BASE1+$ØE     ;Interrupt enable register
          CLI               ;reenable interrupts
          RTS
```

The following routine will intercept the Timer Ø interrupts and count time in hours (HOURS), minutes (MINS), seconds (SECS) and 2Ø millisecond units (UNITS):

```
TIMINT    TYA               ;stack Y register
          PHA
          LDA BASE1+$ØD     ;Interrupt Flag Register
          AND #$4Ø          ;Timer Ø bit on?
          BEQ OTHER         ;Not a timer interrupt
          LDA BASE1+$Ø4     ;Clear interrupt flag
          INC UNITS         ;millisecond counter
          LDA UNITS
```

```
            CMP #$14        ;20 counts completed?
            BMI TIMOK       ;update seconds after 20 ticks
            LDA #$00        ;reset units to 0
            STA UNITS
            INC SECS        ;increment seconds
            LDY #$3C        ;compare to 60
            CPY SECS
            BNE TIMOK
            STA SECS        ;reset seconds to 0
            INC MINS        ;increment minutes
            CPY MINS        ;compare to 60
            BNE TIMOK
            STA MINS        ;reset minutes to 0
            INC HOURS       ;increment hours
TIMOK       PLA             ;unstack Y
            TAY
            LDA $45         ;recover A register
            RTI             ;return from interrupt
```

As an example of using the shift register, we will set up the shift register on the user 6522 to continuously output a bit pattern at a rate governed by Timer 3:

```
            LDA #$10        ;free-running shift register mode
            STA BASE2+$0B   ;Auxiliary Control Register
            LDA RATELO      ;low byte of Timer 3 rate
            STA BASE2+$08
            LDA RATEHI      ;high byte of Timer 3 rate
            STA BASE2+$09
            LDA PATTERN     ;shift register bit pattern
            STA BASE2+$0A   ;start shifting
```

This method could be used to produce musical tones. The pitch of the tone would be controlled by the rate of Timer 3, while the timbre (tone color) would be controlled by the pattern in the shift register. The output on handshake line CB2 could be attenuated to a 0-1 volt range and played on a speaker coupled through an audio amplifier.

Real-Time Clock and Counter/Timer

Connections and Interfacing

All of the signals connected with Timers 2 and 3 on the user 6522 are available on the two 16 pin DIP sockets used for Digital I/O (see Table I[illegible]). Timers 0 and 1 on the dedicated 6522 chip are not as versatile, because their handshake signals are not externally available. Moreover, bits 6 and 7 of port B on the

dedicated 6522 chip are internally connected, and handshake lines CA2 and CB2 are committed for other purposes.

Bear in mind that the current and voltage capabilities of the 6522 chip are quite limited. No input should be outside the range of Ø to 5 volts and output current is limited to one TTL load.

Table I: Cable and Self-Test Adapter Connections

ANALOG INPUT/OUTPUT (1)

Pin #	Signal Description	Self-Test Adapter
1-9	Digital Ground	
10	A/D high input	
11	Analog Ground	
12	A/D low input	
13	D/A high output	
14	-12 Volts out, max. current 50mA	
15	+12 Volts out, max. current 50mA	
16	Digital Ground	

DA to AD+
GRD to AD-

(1) The Analog I/O 16-pin DIP socket is in the upper right corner of the ADALAB interface card. Pin 1 is in the upper right corner of the socket.

DIGITAL INPUT/OUTPUT (2)

Pin #	Port A	Port B
1	Bit 0	Bit 0
16	Bit 1	Bit 1
2	Bit 2	Bit 2
15	Bit 3	Bit 3
3	Bit 4	Bit 4
14	Bit 5	Bit 5
4	Bit 6	Bit 6
13	Bit 7	Bit 7
5,6,7,8	GND	GND
9	CA2	CB2
10	CA1	CB1
11,12	+5V	+5V

(2) The Port B 16-pin DIP socket is in the upper left corner, looking at the component side of the ADALAB interface card. The Port A socket is to the right of the Port B socket. Pin 1 is in the upper right corner of each socket.

Table II: A/D and D/A Converter Digital Codes

Binary	Hexadecimal	Voltage
	A/D CONVERTER	
1110111111111111	$EFFF	A/D Full Scale Positive
1110000000000000	$E000	A/D Zero Volts
1100000000000001	$C001	A/D -1 Least Sign. Bit
1100111111111111	$CFFF	A/D Full Scale Negative
	D/A CONVERTER	
0000000000000000	$0000	D/A Full Scale Positive
0000011111111111	$07FF	D/A Zero Volts
0000100000000000	$0800	D/A -1 Least Sign. Bit
0000111111111111	$0FFF	D/A Full Scale Negative

Table III: Dedicated 6522 Addresses and Functions (1)

Address(2)	Function
BASE1+Ø	High byte of D/A digital value; triggers A/D converter on write
BASE1+1	Low byte of D/A digital value; triggers latching of D/A high byte on read or write
BASE1+2	Data Direction Register B; initialized to $ØF
BASE1+3	Data Direction Register A; initialized to $FF
BASE1+4	Timer Ø Low Byte data (3)
BASE1+5	Timer Ø High Byte data
BASE1+6	Timer Ø Low Byte latched preset value; initialized to $BE
BASE1+7	Timer Ø High Byte latched preset value; initialized to $C7
BASE1+8	Timer 1 Low Byte (3)
BASE1+9	Timer 1 High Byte
BASE1+A	Shift Register (unused)
BASE1+B	Auxiliary Control Register; initialized to $EØ
BASE1+C	Peripheral Control Register; initialized to $8A
BASE1+D	Interrupt Flag Register
BASE1+E	Interrupt Enable Register
BASE1+F	Low byte of D/A digital value; doesn't trigger high byte latch

(1) The initialized values referred to in this table are the result of BRUNning QUICKI/O.

(2) BASE1=$CØØØ+N*1ØØ, where N is the slot number.

(3) Timers Ø and 1 here correspond to timers 1 and 2, respectively, in the VIA data sheets.

Table IV: User 6522 Addresses and Functions(1)

Address(2)	Function
BASE2+Ø	Port B (Output) Data Register
BASE2+1	Port A (Input) Data Register
BASE2+2	Port B Data Direction Register; initialized to $FF
BASE2+2	Port A Data Direction Resister; initialized to $ØØ
BASE2+4	Timer 2 Low Byte data (3)
BASE2+5	Timer 2 High Byte data
BASE2+6	Timer 2 Low Byte latched preset value
BASE2+7	Timer 2 High Byte latched preset value
BASE2+8	Timer 3 Low Byte latched preset value (3)
BASE2+9	Timer 3 High Byte
BASE2+A	Shift register
BASE2+B	Auxiliary Control Register; initialized to $Ø1
BASE2+C	Peripheral Control Register; initialized to $88
BASE2+D	Interrupt Flag Register
BASE2+E	Interrupt Enable Register
BASE2+F	Alternate Port A Data Register--no effect on handshake

(1) The initialized values referred to in this table are the result of BRUNning QUICKI/O.
(2) BASE2=$CØ3Ø+N*$1ØØ, where N is the slot number.
(3) Timers 2 and 3 here correspond to timers 1 and 2, respectively, in the VIA data sheets.

Table V: User 6522 Auxiliary Control Register (BASE2+$ØB)

Bit	State	Result
Ø	Ø	Port A latch is disabled
Ø	1	Port A latches data when CA1 interrupt flag is set
1	Ø	Port B latch is disabled
1	1	Port B latches data when CB1 interrupt flag is set
4,3,2	Ø,Ø,Ø	Shift register is disabled
4,3,2	Ø,Ø,1	Shift in under control of timer 2
4,3,2	Ø,1,Ø	Shift in under control of processor clock
4,3,2	Ø,1,1	Shift in under control of external clock
4,3,2	1,Ø,Ø	Free-running output at rate of timer 2
4,3,2	1,Ø,1	Shift out under control of timer 2
4,3,2	1,1,Ø	Shift out under control of processor clock
4,3,2	1,1,1	Shift out under control of external clock
5	Ø	Timer 3* acts as a one-shot interval timer
5	1	Timer 3 counts pulses on PB6
7,6	Ø,Ø	Timer 2* generates a single interrupt after countdown to Ø
7,6	Ø,1	Timer 2 generates continuous interrupts at free-running rate
7,6	1,Ø	Timer 2 single interrupt mode; also pulses PB7
7,6	1,1	Timer 2 free running mode; also outputs a square wave on PB7.

*Timers 2 and 3 here correspond to timers 1 and 2 in the VIA data sheets.

Table VI: User 6522 Peripheral Control Register(BASE2+$ØC)

Bit	State	Result
Ø	Ø	CA1 interrupt flag set by high to low transition on CA1
Ø	1	CA1 interrupt flag set by low to high transition on CA1
3,2,1	Ø,Ø,Ø	CA2 interrupt flag set by high to low transition on CA2 input
3,2,1	Ø,Ø,1	Like Ø,Ø,Ø mode, but clear by writing logic 1 in interrupt register bit Ø
3,2,1	Ø,1,Ø	CA2 interrupt flag set by low to high transition on CA2 input
3,2,1	Ø,1,1	Like Ø,1,Ø mode, but clear by writing logic 1 in interrupt register bit Ø
3,2,1	1,Ø,Ø	Handshake mode. Set CA2 output low on a read or write to Port A; reset CA2 high with active transition on CA1
3,2,1	1,Ø,1	Pulse output mode. Set CA2 output low for one cycle following a read or write to Port A
3,2,1	1,1,Ø	Manual output mode. Hold CA2 output low
3,2,1	1,1,1	Manual mode; hold CA2 output high
4	Ø	CB1 interrupt flag set by high to low transition on CB1
4	1	CB1 interrupt flag set by low to high transition on CB1
7,6,5	Ø,Ø,Ø	CB2 interrupt flag set by high to low transition on CB2 input
7,6,5	Ø,Ø,1	Like Ø,Ø,Ø mode, but clear by writing logic 1 in interrupt register bit 3
7,6,5	Ø,1,Ø	CB2 interrupt flag set by low to high transition on CB2
7,6,5	Ø,1,1	Like Ø,1,Ø mode, but clear by writing logic 1 into interrupt register bit 3
7,6,5	1,Ø,Ø	Handshake mode. Set CB2 output low on write to Port B; reset CB2 high with active transition on CB1
7,6,5	1,Ø,1	Pulse output mode. Set CB2 output low for one cycle following a write to Port B
7,6,5	1,1,Ø	Manual output mode; hold CB1 output low
7,6,5	1,1,1	Manual mode; hold CB2 output high

Table VII: User 6522 Interrupt Control

INTERRUPT FLAG REGISTER (BASE2+$ØD)

Bit	Set By	Cleared By
Ø	Active transition on CA2	Read or write Port A
1	Active transition on CA1	Read or write Port A
2	Completion of 8 shifts	Read or write shift register
3	Active transition on CB2	Read or write Port B
4	Active transition on CB1	Read or write Port B
5	Time-out of Timer 3*	Read low byte or write high byte
6	Time-out of Timer 2*	Read low byte or write high byte
7	Any of bits Ø-6 set and enabled	Clear bit in flag register or in enable register

INTERRUPT ENABLE REGISTER (BASE2+$ØE)

Bit	Enable	Action
Ø	CA2 Interrupt	Any bit=Ø disables interrupt
1	CA1 Interrupt	Any bit=1 enables interrupt
2	Shift Register Interrupt	
3	CB2 Interrupt	
4	CB1 Interrupt	
5	Timer 3* Interrupt	
6	Timer 2* Interrupt	
7	Writing logic 1 in any bit while bit 7=Ø clears that enable bit Writing logic 1 in any bit while bit 7=1 sets that enable bit	

*Timers 2 and 3 here correspond to timers 1 and 2 in the VIA description sheets.

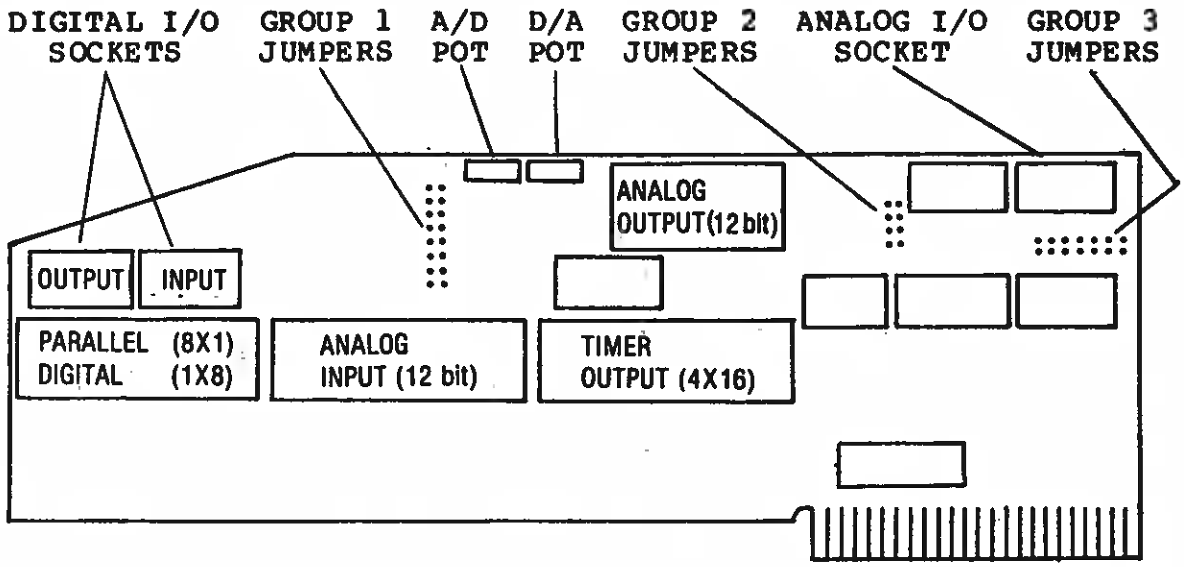


FIGURE 1: Diagram of the ADALAB Interface Card, Connection Sockets, Jumper Select Options and Potentiometer Adjustments

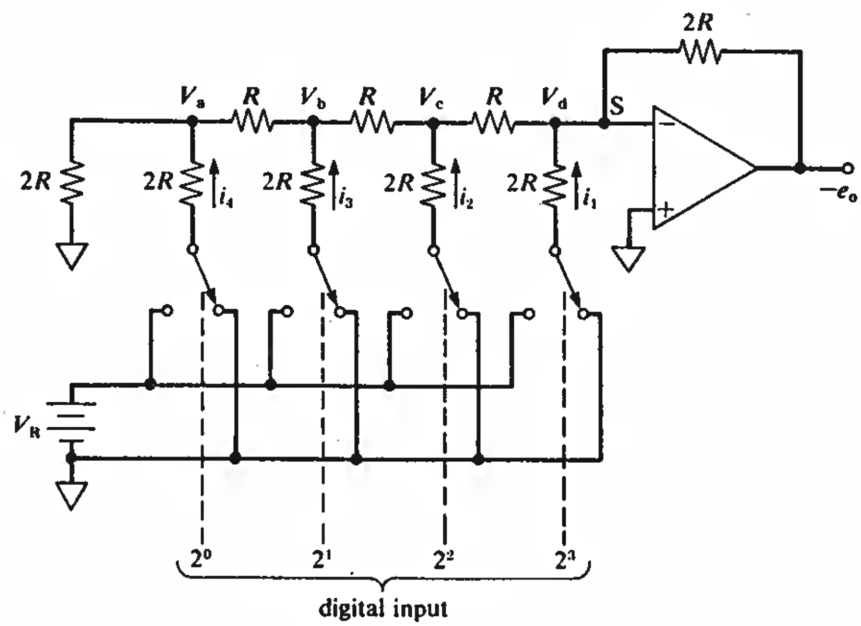


FIGURE 2: Circuit Diagram for a 4-bit Digital to Analog Converter. For more details and a circuit analysis, refer to H. V. Malmstadt and C. G. Enke, Digital Electronics for Scientists (W. A. Benjamin, Inc., New York, 1969), pp. 333-335.

MCS6522

MCS6522
Versatile Interface Adapter (VIA)

- **Completely Static**
- **Fully TTL Compatible**
- **CMOS Compatible Peripheral Control Lines**
- **8-Bit Bidirectional Data/Control Transfer**
- **2 Powerful Interval Timers**
- **Shift Register for Serial/Parallel and Parallel/Serial Transfers**
- **Input Data Latching on Peripheral Ports**
- **Fully Automatic Handshake**
- **Independent Interrupt Control**
- **Single +5V Supply**

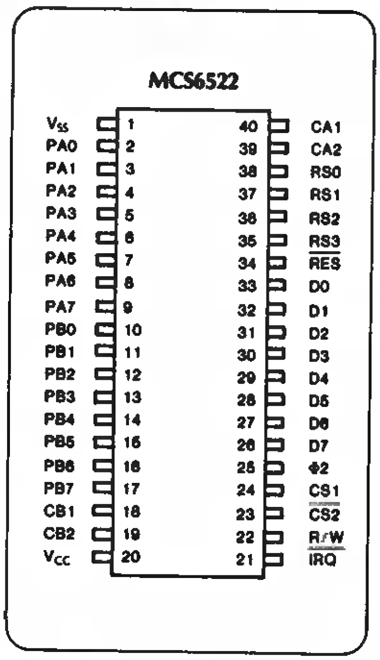
DESCRIPTION

The MCS6522 Versatile Interface Adapter (VIA) provides all of the capability of the MCS6520 Peripheral Adapter. In addition, it offers a pair of powerful interval timers, a serial-to-parallel/parallel-to-serial shift register and input data latching on the peripheral ports. Expanded handshaking capability over that of the MCS6520 allows control of bidirectional data transfers between VIAs in a multiple processor system.

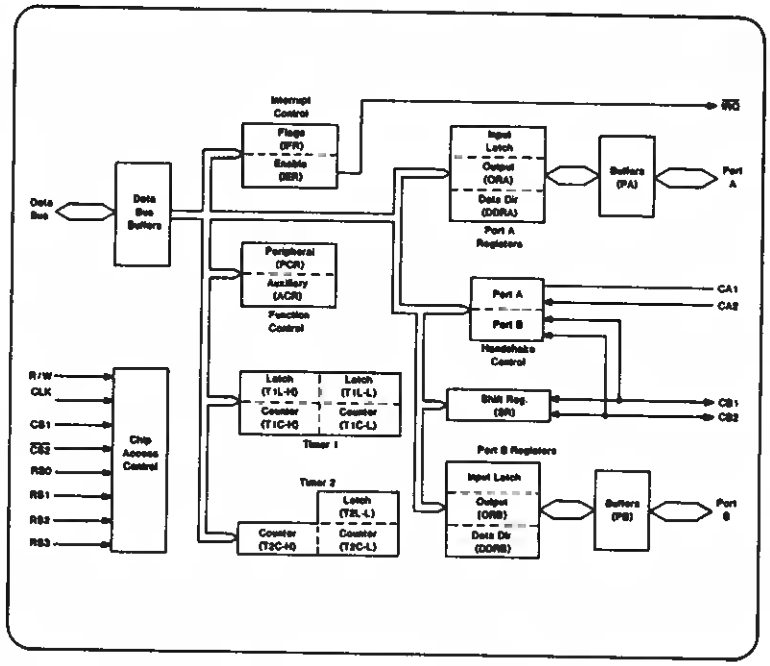
Control of peripheral devices is handled primarily through two 8-bit bidirectional ports. Each line in these ports can be programmed to act as either an input or an output. Serveral peripheral I/O lines can also be controlled directly from the MCS6522's internal interval timer, permitting the generation of programmable-frequency square waves and for counting pulses generated externally. Internal registers are organized into an interrupt flag register, an interrupt enable register and a pair of function control registers. This permits easy control of the many features of the device.

4

PIN CONFIGURATION



BLOCK DIAGRAM



INTERFACE TO THE PROCESSOR

This section contains a description of the buses and control lines which are used to interface the MCS6522 to the system processor.

Phase Two Clock (Φ2). Data transfers between the MCS6522 and the system processor take place only while the Phase Two Clock is high. In addition, Φ2 acts as the time base for the various timers and shift registers on the chip.

Chip Select Lines (CS1, $\overline{CS2}$). The two chip select inputs are normally connected to processor address lines either directly or through decoding. The selected MCS6522 register will be accessed when CS1 is high and $\overline{CS2}$ is low.

Register Select Lines (RS0, RS1, RS2, RS3). The four Register select lines are normally connected to the processor address bus lines to allow the processor to select the internal MCS6522 register which is to be accessed. The sixteen possible combinations access the registers shown in Table 1.

Table 1. Register Select Line Definitions

RS3	RS2	RS1	RS0	Register	Remarks
L	L	L	L	ORB	
L	L	L	H	ORA	Controls Handshake
L	L	H	L	DDRB	
L	L	H	H	DDRA	
L	H	L	L	T1L-L T1C-L	Write Latch Read Counter
L	H	L	H	T1C-H	Trigger T1L-L/ T1C-L Transf.
L	H	H	L	T1L-L	
L	H	H	H	T1L-H	
H	L	L	L	T2L-L T2C-L	Write Latch Read Counter
H	L	L	H	T2C-H	Triggers T2L-L/ T2C-L Transfer
H	L	H	L	SR	
H	L	H	H	ACR	
H	H	L	L	PCR	
H	H	L	H	IFR	
H	H	H	L	IER	
H	H	H	H	ORA	No Effect on Handshake

Read/Write Line (R/W). The direction of data transfers between the MCS6522 and the system processor is controlled by the R/W line. If R/W is low, data will be transferred out of the processor into the selected MCS6522 register (write operation). If R/W is high and the chip is selected, data will be transferred out of the MCS6522 to the data bus (read operation).

Data Bus (DB0 - DB7). The 8 bi-directional data bus lines are used to transfer data between the MCS6522 and the system processor. The internal drivers will remain in the high-impedance state except when the chip is selected (CS1 = 1, $\overline{CS2}$ = 0), Read/Write is high and the Phase Two Clock is high. At this time, the contents of the selected register are placed on the data bus. When the chip is selected, with Read/Write low and Φ2 = 1, the data on the data bus will be transferred into the selected MCS6522 register.

Reset ($\overline{RES}$). The Reset input clears all internal registers (except T1, T2, and SR) to logic 0. This places all peripheral interface lines in the input state, disables the timers, shift register, and interrupts from the chip.

Interrupt Request ($\overline{IRQ}$). The Interrupt Request output goes low whenever an internal interrupt flag is set and the correspondeing interrupt enable bit is a logic 1. This output is "open drain" to allow the interrupt request signal to be wire-ORed with other equivalent signals in the system.

INTERFACE TO THE PERIPHERAL. This section contains a brief description of the buses and control lines used to drive peripheral devices under control of the MCS6522 registers.

Peripheral A Port (PA0 - PA7). The Peripheral A port consists of 8 lines which can be individually programmed to act as input or output under control of a Data Direction Register. The polarity of output pins is controlled by an Output Register and input data can be latched into an internal register under control of the CA1 line. All of these modes of operation are controlled by the system processor through the internal control registers. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode.

Peripheral A Control Lines (CA1, CA2). The two peripheral A control lines act as interrupt inputs or as handshake outputs. Each line controls an internal interrupt flag with a corresponding interrupt enable bit. In addition, CA1 controls the latching of data on Peripheral A Port input lines. The various modes of operation are controlled by the system processor through the internal control registers. CA1 is a high-impedance input only while CA2 represents one standard TTL load in the input mode. CA2 will drive one standard TTL load in the output mode.

Peripheral B Port (PB0 - PB7). The Peripheral B port consists of 8 bi-directional lines controlled by an output register and a Data Direction Register in much the same manner as the PA port. In addition, the polarity of the PB7 output signal can be controlled by one of the interval timers while the second timer can be programmed to count pulses on the PB6 pin. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode.

Peripheral B Control Lines (CB1, CB2). The Peripheral B control lines act as interrupt inputs or as handshake outputs. As with CA1 and CA2, each line controls an interrupt flag with a corresponding interrupt enable bit. In addition, these lines act as a serial port under control of the Shift Register. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode.

OPERATION

This section contain a discussion of the various blocks of logic shown in the block diagram. In addition, the internal operation of the MCS6522 is described in detail.

Chip Access Control.

The Chip Access Control contains the necessary logic to detect the chip select condition and to decode the Register Select inputs to allow access to the desired register. In addition, the R/W and $\Phi 2$ signals are utilized to control the direction and timing of data transfers. When writing into the MCS6522, data is first latched into a data input register during $\Phi 2$. Data is then transferred into the desired internal register during $\overline{\Phi 2}$ - $\overline{\text{Chip Select}}$. This allows the peripheral I/O line to change without "glitching." When the processor reads the MCS6522, data is transferred from the desired internal register directly onto the Data Bus during $\Phi 2$.

Port A Registers, Port B Registers

Three registers are used in accessing each of the 8-bit peripheral ports. Each port has a Data Direction Register (DDRA, DDRB) for specifying whether the peripheral pins are to act as inputs or outputs. A 0 in a bit of the Data Direction Register causes the corresponding peripheral pin to act as an input. A 1 causes the pin to act as an output.

Each peripheral pin is also controlled by a bit in the Output Register (ORA, ORB) and an Input Register (IRA,IRB). When the pin is programmed to act as an output, the voltage on the pin is controlled by the corresponding bit of the Output Register. A 1 in the Output Register causes the pin to go high, and a 0 causes the pin to go low.

Reading a peripheral port causes the contents of the Input Register (IRA, IRB) to be transferred onto the Data Bus. With input latching disabled, IRA will always reflect the data on the PA pins. With input latching enabled, IRA will reflect the contents of the Port A prior to setting the CA1 Interrupt Flag (IFR1) by an active transition on CA1.

The IRB register operates in a similar manner. However, for output pins, the corresponding IRB bit will reflect the contents of the Output Register bit instead of the actual pin. This allows proper data to be read into the processor if the output pin is not allowed to go to full voltage. With input latching enabled on Port B, setting CB1 interrupt flag will cause IRB to latch this combination of input data and ORB data until the interrupt flag is cleared.

Handshake Control

Th MCS6522 allows positive control of data transfers between the system processor and peripheral devices through the operation of "handshake" lines. Port A lines (CA1, CA2) handshake data on both a read and a write operation while the Port B lines (CB1, CB2) handshake on a write operation only.

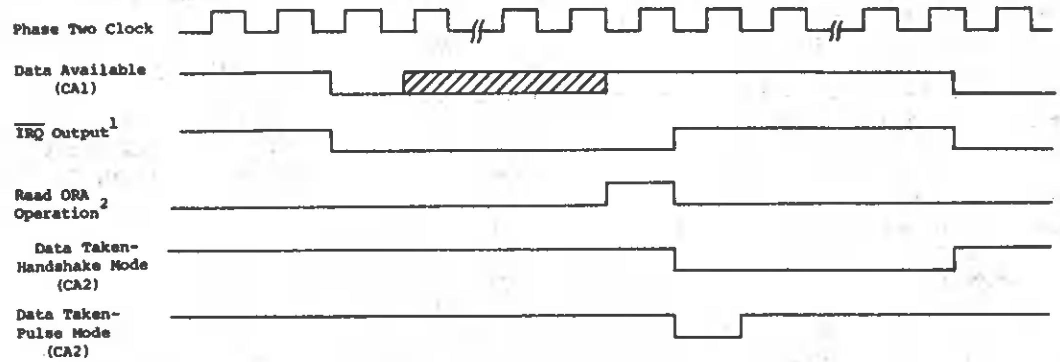
Read Handshake. Positive control of data transfers from peripheral devices into the system processor can be accomplished using "Read" handshaking. In this case, the peripheral device must generate "Data Ready" to signal the processor that valid data is present on the peripheral port. This signal normally interrupts the processor, which then reads the data, causing generation of a "Data Taken" signal. The peripheral device responds by making new data available. This process continues until the data transfer is complete.

In the MCS6522, automatic "Read" handshaking is possible on the Peripheral A port only. The CA1 interrupt input pin accepts the "Data Ready" signal and CA2 generates the "Data Taken" signal. The Data Ready signal will set an internal flag which may either interrupt the processor or be polled by software. The Data Taken signal can be either a pulse or a DC level which is set low by the system processor and cleared by the Data Ready signal. These options are shown in Figure 1 which illustrates the normal Read handshaking sequence.

Write Handshake. The sequence of operations which allows handshaking data from the system processor to a peripheral device is very similar to that described for Read Handshaking. However, for "Write" handshaking, the processor must generate the "Data Ready" signal (through the MCS6522) and the peripheral device must respond with the "Data Taken" signal. This can be accomplished on both the PA port and the PB port on the MCS6522. CA2 or CB2 acts as a Data Ready output in either the DC level or pulse mode and CA1 or CB1 accepts the "Data Taken" signal from the peripheral device, setting the interrupt flag and clearing the "Data Ready" output. This sequence is shown in Figure 2.

4

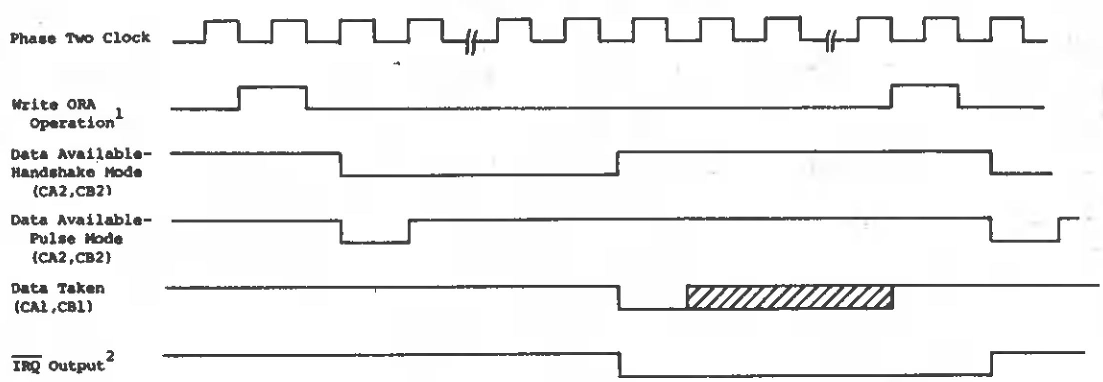
READ HANDSHAKE TIMING SEQUENCE



NOTES
1. Signals "data available" to the system processor.
2. R/W = 1, $\overline{CS2}$ = 0, CS1 = 1, RS3 = 0, RS2 = 0, RS1 = 0, RS0 = 1.

Figure 1.

WRITE HANDSHAKE TIMING SEQUENCE



NOTES
1. R/W = 0, $\overline{CS2}$ = 1, RS3 = 0, RS2 = 0, RS1 = 0, RS0 = 1.
2. Signals "data taken" to the system processor.

Figure 2.

Timer 1 (T1)

Interval Timer T1 consists of two 8-bit latches and a 16-bit counter. The latches are used to store data to be loaded into the counter. After loading, the counter decrements at system clock rate. Upon reaching zero, an interrupt flag will be set, and $\overline{IRQ}$ will go low. The timer will then disable any further interrupts, or automatically transfer the contents of the latches into the counter and continue to decrement. In addition, the timer can be instructed to invert the output signal on a peripheral pin each time it "times-out". Each of these modes is discussed separately below.

Writing T1. Operations which take place when writing to each of the four T1 addresses are shown in Table 2.

Table 2. Writing to T1 Registers

RS3	RS2	RS1	RS0	Operation (R/W = L)
L	H	L	L	Write into low order latch.
L	H	L	H	Write into high order latch. Write into high order counter. Transfer low order latch into low order counter. Reset T1 interrupt flag.
L	H	H	L	Write low order latch.
L	H	H	H	Write high order latch. Reset T1 interrupt flag.

Note that the processor does not write directly into the low order counter (T1C-L). Instead, this half of the counter is loaded automatically from the low order latch when the processor writes into the high order counter.

The second set of addresses allows the processor to write into the latch register without affecting the count-down in progress. This is discussed in detail below.

Reading T1 Registers. For reading the Timer 1 registers, the four addresses relate directly to the four registers as shown in Table 3.

Table 3. Reading T1 Registers

RS3	RS2	RS1	RS0	Operation (R/W = H)
L	H	L	L	Read T1 low order counter. Reset T1 interrupt flag.
L	H	L	H	Read T1 high order counter.
L	H	H	L	Read T1 low order latch.
L	H	H	H	Read T1 high order latch.

Timer 1 Operating Modes. Two bits are provided in the Auxiliary Control Register to allow selection of the T1 operating modes. These bits and the four possible mòdes are shown in Table 4.

Table 4. T1 Operating Modes

ACR7 Output Enable	ACR6 "Free-Run" Enable	Mode
0	0	Generate a single time-out interrupt each time T1 is loaded. PB7 disabled.
0	1	Generate continuous interrupts. PB7 disabled.
1	0	Generate a single interrupt and an output pulse on PB7 for each T1 load operation.
1	1	Generate continouos interrupts and a square wave output on PB7.

One-Shot Mode. The interval timer one-shot mode allows generation of a single interrupt for each timer load operation. As with any interval timer, the delay between the "write T1C-H" operation and generation of the processor interrupt is a direct function of the data loaded into the timing counter. In addition to generating a single interrupt, Timer 1 can be programmed to produce a single negative pulse on the PB7 peripheral pin. With the output enabled (ACR7 = 1) a "write T1C-H" operation will cause PB7 to go low. PB7 will return high when Timer 1 times out. The result is a single programmable width pulse.

NOTE

PB7 will act as an output if DDRB7 = 1 or if ACR7 = 1. However, if both DDRB7 and ACR7 are logic 1, PB7 will be controlled from Timer 1 and ORB7 will have no effect on the pin

In the one-shot mode, writing into the high order latch has no effect on the operation of T1. However, it will be necessary to assure that the low order latch contains the proper data before initiating the count-down with a "write T1C-H" operation. When the processor writes into the high-order counter, the T1 interrupt flag will be cleared, the contents of the low-order latch will be transferred into the low-order counter, and the timer will begin to decrement at system clock rate. If the PB7 output is enabled, this signal will go low on the phase two following the write operation. When the counter reaches zero, the T1 interrupt flag will be set, the $\overline{IRQ}$ pin will go low (interrupt enabled), and the signal on PB7 will go high. At this time the counter will continue to decrement at system clock rate. This allows the system processor to read the contents of the counter to determine the time since interrupt. However, the T1 interrupt flag cannot be set again unless it has been cleared.

4

Free-Running Mode. The most important advantages associated with the latches in T1 are the ability to produce a continuous series of evenly spaced interrupts and the ability to produce a square wave on PB7 whose frequency is not affected by variations in the processor interrupt response time. This is accomplished in the "free-running" mode.

In the free-running mode (ACR6 = 1), the interrupt flag is set and the signal on PB7 is inverted each time the counter reaches zero. However, instead of contining to decrement from zero after a time-out the timer automatically transfers the contents of the latch into the counter (16 bits) and continues to decrement from there. The interrupt flag can be cleared by writing T1C-H, by reading T1C-L, or by writing directly into the flag as described below. However, it is not necessary to rewrite the timer to enable setting the interrupt flag on the next time-out

All interval timers in the MCS6500 family devices are "re-triggerable." Rewriting the counter will always re-initialize the time-out period. In fact, the time-out can be prevented completely if the processor continues to rewrite the timer before it reaches zero. Timer 1 will operate in this manner if the processor writes into the high order counter (T1C-H). However, by loading the latches only, the processor can access the timer during each down-counting operation without affecting the time-out in process. Instead, the data loaded into the latches will determine the length of the next time-out period.

Timer 2 (T2)
Timer 2 operates as an interval timer (in the "one-shot" mode only), or as a counter for negative pulses on the PB6 peripheral pin. A single control bit is provided in the Auxiliary Control Register to select between these two modes. This timer is comprised of a "write only" low-order latch (T2L-L), a "read-only" low-order counter and a read/write high-order counter. The counter registers act as a 16-bit counter which decrements at Φ2 rate.

Timer 2 addressing is summarized in Table 5.

Table 5. T2 Addressing

RS3	RS2	RS1	RS0	R/W = 0	R/W = 1
H	L	L	L	Write T2L-L	Read T2-L Clear Interrupt flag
H	L	L	H	Write T2C-H Transfer T2L-L to T2C-L Clear Interrupt flag	Read T2C-H

T2 Interval Timer Mode. As an interval timer, T2 operates in the "one shot" mode similar to T1. In this mode, T2 provides a single interrupt for each "write T2C-H" operation. After timing out, the counter will continue to decrement. However, setting of the interrupt flag will be disabled after initial time-out so that it will not be set by the counter continuing to decrement through zero. The processor must rewrite T2C-H to enable setting of the interrupt flag. The interrupt flag is cleared by reading T2C-L or by writing T2C-H

T2 Pulse-Counting Mode. In the pulse-counting mode, T2 serves primarily to count a predetermined number of negative-going pulses on PB6. This is accomplished by first loading a number into T2. Writing into T2C-H clears the interrupt flag and allows the counter to decrement each time a pulse is applied to PB6. The interrupt flag will be set when T2 reaches zero. At this time the counter will continue to decrement with each pulse on PB6. However, it is necessaryu to rewrite T2C-H to allow the interrupt flag to set on subquent down-counting operations. The pulse must be low on the leading edge Φ2.

Shift Register (SR)
The Shift Register (SR) performs serial data transfers into and out of the CB2 pin under control of an internal modulo-8 counter. Shift pulses can be applied to the CB1 pin from an external source or, with the proper mode selection, shift pulses generated internally will appear on the CB1 pin for controlling shifting in external devices. The control bits which allow control of the various shift register operating modes are located in the Auxiliary Control Register. These bits can be set and cleared by the system processor to select one of the operating modes.

SR Input Modes. Bit 4 of the Auxiliary Control Register selects the input or output modes. There are three input modes and four output modes, differing primarily in the source of the pulses which control the shifting operation. With ACR4 = 0 the input modes are selected by ACR3 and ACR2 as shown in Table 6.

Table 6. SR Input Mode Selection

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled
0	0	1	Shift in under control of Timer 2
0	1	0	Shift in at System Clock Rate
0	1	1	Shift in under control of external input pulses on CB1?

SR Output Modes. The four Shift Register Output Modes are selected by setting the Input/Output Control Bit (ACR4) to a logic 1 and then selecting the specific output mode with ACR3 and ACR2. In each of these modes the Shift Register shifts data out of Bit 7 to the CB2 pin. At the same time the contents of Bit 7 are shifted back into Bit 0. As in the input modes, CB1 is used either as an output to provide shifting pulses out or as an input to allow shifting from an external pulse. The four modes are shown in Table 7.

Table 7. SR Output Mode Selection

ACR4	ACR3	ACR2	Mode
1	0	0	Shift out – Free-running mode. Shift rate controlled by T2.
1	0	1	Shift out – Shift rate controlled by T2. Shift pulses generated on CB1.
1	1	0	Shift out at system clock rate.
1	1	1	Shift out under control of an external pulse. on CB1

Interrupt Control
Controlling interrupts within the MCS6522 involves three principal operations. These are flagging the interrupts, enabling interrupts and signalling to the processor that an active interrupt exists. Interrupt flags are set by interrupting conditions which exist within the chip or on inputs to the chip. These flags normally remain set until the interrupt has been serviced. To determine the source of an interrupt, the microprocessor must examine these flags in order from highest to lowest priority. This is accomplished by

reading the flag register into the processor accumulator, shifting this register either right or left and then using conditional branch instructions to detect an active interrupt.

Associated with each interrupt flag is an interrupt enable bit. This bit can be set or cleared by the processor to enable interrupting the processor from the corresponding interrupt flag. If an interrupt flag is set to a logic 1 by an interrupting condition, and the corresponding interrupt enable bit is set to a 1, the Interrupt Request Output ($\overline{IRQ}$) will go low. $\overline{IRQ}$ is an "open-collector" output which can be "wire-ORed" with other devices in the system to interrupt the processor.

In the MCS6522, all the interrupt flags are contained in one register (see Table 8). In addition, Bit 7 of this register will be read as a logic 1 when an interrupt exists within the chip. This allows convenient polling of several devices within a system to locate the source of an interrupt.

Table 8. Interrupt Flags

	7	6	5	4	3	2	1	0
Interrupt Flag Register	IRQ	T1	T2	CB1	CB2	SR	CA1	CA2
Interrupt Enable Register	Set/clear control	T1	T2	CB1	CB2	SR	CA1	CA2

Interrupt Flag Register (IFR). The IFR is a read/bit-clear register. When the proper chip select and register signals are applied to the chip, the contents of this register are placed on the data bus. Bit 7 indicates the status of the $\overline{IRQ}$ output. This bit corresponds to the logic function: $IRQ = IFR6 \times IER6 + IFR5 \times IER5 + IFR4 \times IER4 + IFR3 \times IER3 + IFR2 \times IER2 + IFR1 \times IER1 + IFR0 \times IER0$, where $\times$ = logical AND, $+$ = logical OR.

Bits six through zero are latches which are set and cleared as shown in Table 9.

IFR Bit 7 is not a flag. Therefore, this bit is not directly cleared by writing a logic 1 into it. It can only be cleared by clearing all the flags in the register or by disabling all the active interrupts.

Interrupt Enable Register (IER). For each interrupt flag in IFR, there is a corresponding bit in the Interrupt Enable Register. The system processor can set or clear selected bits in this register to facilitate controlling individual interrupts without affecting others. This is accomplished by writing to address 1110 (IER address). If Bit 7 of the data placed on the system data bus during this write operation is a 0, each 1 in Bits 6 through 0 clears the corresponding bit in the Interrupt Enable Register. For each zero in bits 6 through 0, the corresponding bit is unaffected.

Table 9. Bits 6–0 of IFR

Bit #	Set by	Cleared by
0	Active transition of the signal on the CA2 pin.	Reading or writing the A Port Output Register (ORA) using address 0001.
1	Active transition of the signal on the CA1 pin.	Reading or writing the A Port Output Reigster (ORA) using address 0001.
2	Completion of eight shifts	Reading or writing the Shift Register.
3	Active transition of the signal on the CB2 pin.	Reading or writing the B Port Output Register.
4	Active transition of the signal on the CB1 pin.	Reading or writing the B Port Output Register
5	Time-out of Timer 2.	Reading T2 low order counter. Writing T2 high order counter.
6	Time-out of Timer 1.	Reading T1 low order counter. Writing T1 high order latch.

Setting selected bits in the IER is accomplished by writing to the same address with Bit 7 in the data word set to a logic 1. In this case, each 1 in Bits 6 through 0 will set the corresponding bit. For each zero, the corresponding bit will be unaffected. This individual control of the setting and clearing operations allows convenient control of interrupts during system operation.

In addition to setting and clearing IER bits, the processor can read the contents of this register by placing the proper address on the register select and chip select inputs with the R/W line high. Bit 7 will be read as a logic 0.

Function Control

Control of the various functions and operating modes within the MCS6522 is accomplished primarily through two registers, the Peripheral Control Register (PCR), and the Auxiliary Control Register (ACR). The PCR is used primarily to select the operating mode for the four peripheral control pins. The Auxiliary Control Register selects the operating mode for the interval timers (T1, T2), and the Shift Register (SR).

Peripheral Control Register (PCR). The Peripheral Control Register is organized as shown in Figure 3.

Figure 3. PCR Organization

Bit #	7	6	5	4	3	2	1	0
Function	CB2 Control			CB1 Control	CA2 Control			CA1 Control

4

Each of these functions is discussed in detail below.

1. CA1 Control

Bit 0 of the PCR selects the active transition of the input signal applied to the CA1 interrupt input pin. If this bit is a logic 0, the CA1 interrupt flag will be set by a negative transition (high to low) of the signal on the CA1 pin. If PCR0 is a logic 1, the flag will be set by a positive transition.

2. CA2 Control

The CA2 pin can be programmed to act as an interrupt input or as a peripheral control output. As an input, CA2 operates in two modes, differing primarily in the methods available for resetting the interrupt flag. Each of these two input modes can operate with either a positive or a negative active transition as described above for CA1.

In the output mode, the CA2 pin combines the operations performed on the CA2 and CB2 pins of the MCS6520. This added flexibility allows the processor to perform a normal "write" handshaking in a system which uses CB1 and CB2 for the serial operations described above. The CA2 operating modes are selected as shown in Table 10.

In the independent input mode, writing or reading the ORA register has no effect on the CA2 interrupt flag. This flag must be cleared by writing a logic 1 into the appropriate IFR bit. This mode allows the processor to handle interrupts which are independent of any operations taking place on the peripheral I/O ports.

3. CB1 Control

Control of the active transition of the CB1 input signal operates in exactly the same manner as that described above for CA1. If the Shift Register function has been enabled, CB1 will act as an input or output for the shift register clock signals. In this mode the CB1 interrupt flag will still respond to the selected transition of the signal on the CB1 pin.

Table 10. CA2 Operating Mode Selection

PCR3	PCR2	PCR1	Mode
0	0	0	**Input mode.** Set CA2 interrupt flag (IFR0) on a negative transition of the input signal. Clear IFR0 on a read or write of the Peripheral A Output Register.
0	0	1	**Independent interrupt input mode.** Set IFR0 on a negative transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag.
0	1	0	**Input mode.** Set CA2 interrupt flag on a positive transition of the CA2 input signal. Clear the IFR0 with a read or write of the Peripheral A Output Register.
0	1	1	**Independent interrupt input mode.** Set IFR0 on a positive transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag.
1	0	0	**Handshake output mode.** Set CA2 output low on a read or write of the Peripheral A Output Register. Reset CA2 high with an active transition on CA1.
1	0	1	**Pulse Output mode.** CA2 goes low for one cycle following a read or write of the Peripheral A Output Register.
1	1	0	**Manual output mode.** The CA2 output is held low in this mode.
1	1	1	**Manual output mode.** The CA2 output is held high in this mode.

4. CB2 Control
With the serial port disabled, operation of the CB2 pin is a function of the three high-order bits of the PCR. The CB2 modes are very similar to those described previously for CA2, and are selected as shown in Table 11.

Table 11. CB2 Operating Mode Selection

PCR7	PCR6	PCR5	Mode
0	0	0	**Interrupt input mode.** Set CB2 interrupt flag (IFR3) on a negative transition of the CB2 input signal. Clear IFR3 on a read or write of the Peripheral B Output Register.
0	0	1	**Independent interrupt input mode.** Set IFR3 on a negative transition of the CB2 input signal. Reading or writing ORB does not clear the CA2 interrupt flag.
0	1	0	**Input mode.** Set CB2 interrupt flag on a positive transition of the CB2 input signal. Clear the CB2 interrupt flag on a read or write of ORB.
0	1	1	**Independent input mode.** Set IFR3 on a positive transition of the CB2 input signal. Reading or writing ORB does not clear the CB2 interrupt flag.
1	0	0	**Handshake output mode.** Set CB2 low on a write ORB operation. Reset CB2 high with an active transition of the CB1 input signal.
1	0	1	**Pulse output mode.** Set CB2 low for one cycle following a write ORB operation.
1	1	0	**Manual output mode.** The CB2 output is held low in this mode.
1	1	1	**Manual output mode.** The CB2 output is held high in this mode.

Auxiliary Control Register (ACR). Many of the functions in the Auxiliary Control Register have been discussed previously. However, a summary of this register is presented here as a convenient reference. ARC organization is shown in Figure 4.

Figure 4. ACR Organization

Bit #	7	6	5	4	3	2	1	0
Function	T1 Control		T2 Control	Shift Register Control			PB Latch Enable	PA Latch Enable

Each of these functions is described in detail below.

1. PA Latch Enable
The MCS6522 provides input latching on both the PS and PB ports. In this mode, the data present on the peripheral A input pins will be latched within the chip when the CA1 interrupt flag is set. Reading the PA port will result in these latches being transferred into the processor. As long as the CA1 interrupt flag is set, the data on the peripheral pins can change without affecting the data in the latches. This input latching can be used with any of the CA2 input or output modes.

It is important to note that on the PA port, the processor always reads the data on the peripheral pins (as reflected in the latches). For output pins, the processor still reads the latches. This may or may not reflect the data currently in the ORA. Proper system operation requires careful planning on the part of the system designer if input latching is combined with output pins on the peripheral ports.

Input latching is enabled by setting Bit 0 in the Auxiliary Control Register to a logic 1. As long as this bit is a 0, the latches will directly reflect the data on the pins.

2. PB Latch Enable
Input latching on the PB port is controlled in the same manner as that described for the PS port. However, with the Peripheral B port, the input latch will store either the voltage on the pin or the contents of the Output Register (ORB), depending on whether the pin is programmed to act as an input or an output. As with the PA port, the processor always reads the input latches.

3. Shift Register (SR) Control
The Shift Register operating mode is selected as shown in Table 12.

Table 12. SR Operating Mode Selection

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled.
0	0	1	Shift in Under Control of Timer 2.
0	1	0	Shift in Under Control of System Clock.
0	1	1	Shift in Under Control of External Clock Pulses.
1	0	0	Free-running Output at Rate Determined by Timer 2.
1	0	1	Shift Out Under Control of Timer 2.
1	1	0	Shift Out Under Control of the System Clock.
1	1	1	Shift Out Under Control of External Clock Pulses.

4

4. T2 Control

If ACR5 = 0, T2 acts as an interval timer in the one-shot mode. If ACR5 = 1, Timer 2 acts to count a predetermined number of pulses on pin PB6.

5. T1 Control

Timer 1 operates in the one-shot or free-running mode with the PB7 output control enabled or disabled. These modes are selected as shown in Table 13.

Table 13. T1 Mode Selection

ACR7	ACR6	Mode
0	0	One-shot Mode- Output to PB7 Disabled.
0	1	Free-running Mode- Output to PB7 Disabled.
1	0	One-shot Mode- Output to PB7 Enabled.
1	1	Free-running Mode. Output to PB7 Enabled.

ABSOLUTE MAXIMUM RATINGS

Parameter	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	Vdc
Input Voltage	V_{in}	−0.3 to +7.0	Vdc
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

CAUTION

This device contains circuitry to protect the inputs against damage due to high static voltages. However, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages.

DC CHARACTERISTICS V_{CC} = 5.0 V ± 5%, V_{SS} = 0, T_A = 0 to +70 °C (unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
V_{IH}	Input High Voltage (normal operation)	+2.4		V_{CC}	Vdc	
V_{IL}	Input Low Voltage (normal operation)	−0.3		+0.4	Vdc	
I_{IN}	Input Leakage Current		±1.0	±2.5	µAdc	V_{in} = 0 to 5 Vdc R/W, $\overline{RES}$, RS0, RS1, RS2, RS3, CS1, $\overline{CS2}$, CA1, Φ 2
I_{TSI}	Off-State Input Current		±2.0	±10	µAdc	V_{in} = .4 to 2.4 V V_{CC} = Max, D0 to D7
I_{IH}	Input High Current	−100	−250		µAdc	V_{IH} = 2.4 V PA0 - PA7, CA2, PB0 - PB7, CB1, CB2
I_{IL}	Input Low Current		−1.0	−1.6	mAdc	V_{IL} = 0.4 Vdc PA0 - PA7, CA2, PB0 - PB7, CB1, CB2
V_{CH}	Output High Voltage	2.4			Vdc	V_{CC} = min, I_{load} = −100 µAdc PA0 - PA7, CA2, PB0 - PB7, CB1, CB2
V_{OL}	Output Low Voltage			+0.4	Vdc	V_{CC} = min, I_{load} = 1.6 mAdc
I_{OH}	Output High Current (sourcing)	−100 −3.0	−1000 −5.0		µAdc mAdc	V_{OH} = 2.4 V V_{OH} = 1.5 V, PB0 - PB7, CB1, CB2
I_{OL}	Output Low Current (sinking)	1.6			mAdc	V_{OL} = 0.4 Vdc.
I_{off}	Output Leakage Current (off state)		1.0	10	µAdc	$\overline{IRQ}$
C_{in}	Input Capacitance			7.0 10 20	pF pF pF	T_A = 25 °C, f = 1 Mhz R/W, $\overline{RES}$, RE0, RS1, RS2, RS3, CS1, CS2 D0 - D7, PA0 - PA7, CA2, PB0 - PB7, CB1, CB2 Φ 2 input
C_{out}	Ouput Capacitance			10	pF	T_A = 25 °C, f = 1 Mhz
P_d	Power Dissipation			1000	MW	

READ TIMING CHARACTERISTICS

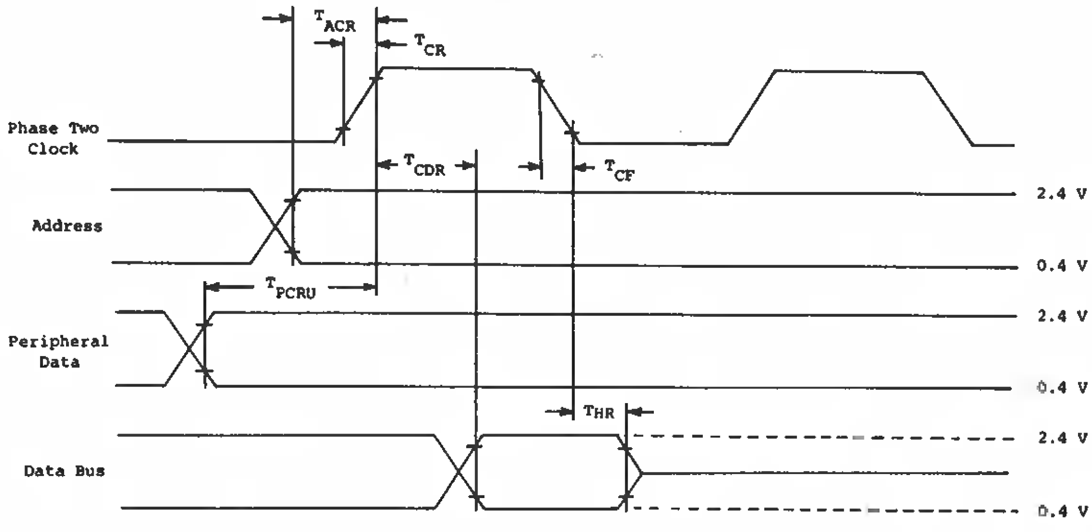


Figure 21.

WRITE TIMING CHARACTERISTICS

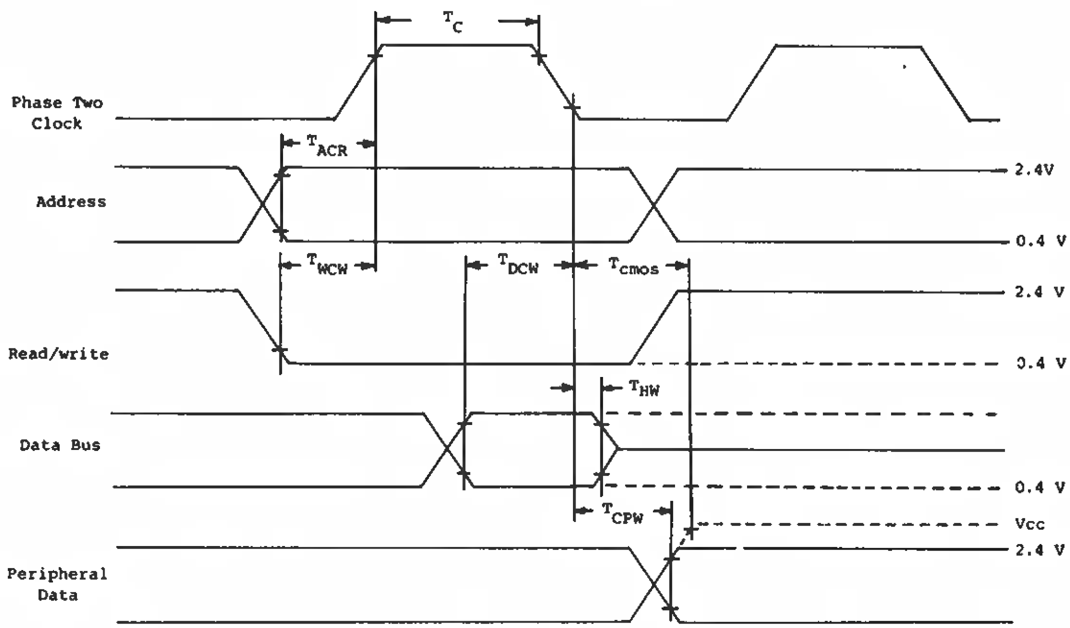


Figure 22.

I/O TIMING CHARACTERISTICS

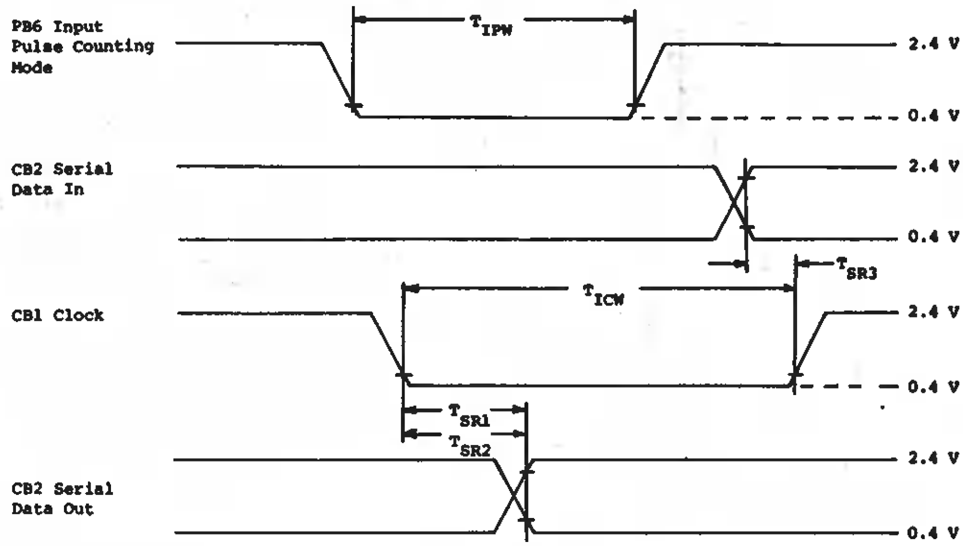


Figure 23.

AC CHARACTERISTICS TA = 0°C to +17°C, V_{CC} = 5V ± 5% (unless otherwise specified)

Symbol	Parameter	Min	Typ	Max	Unit
	READ CYCLE (Figure 22, loading 130 pF and one TTL load)				
T_{ACR}	Delay Time, Address Valid to Clock Positive Transition	180			nS
T_{CDR}	Delay Time, Clock Positive Transition to Data Valid on Bus			395	nS
T_{PCR}	Peripheral Data Setup Time	300			nS
T_{HR}	Data Bus Hold Time	10			nS
T_{RC} T_{RF}	Rise and Fall Time For Clock Input			25	nS
	WRITE CYCLE (Figure 22)				
T_C	Enable Pulse Width	0.47		25	
T_{ACW}	Delay Time, Address Valid to Clock Positive Transition	180			nS
T_{DCW}	Delay Time, Data Valid to Clock Negative Transition	300			nS
T_{WCW}	Delay Time, Read/Write Negative Transition to Clock Positive Transition	180			nS
T_{HW}	Data Bus Hold Time	10			nS
T_{CPW}	Delay Time, Enable Negative Transition to Peripheral Data Valid			1.0	µS
T_{CMOS}	Delay Time, Clock Negative Transition to Peripheral Data Valid CMOS (V_{CC} − 30%)			2.0	µS

Peripheral Interface Characteristics

Symbol	Parameter	Min	Typ	Max	Unit
T_{RF}	Rise and Fall Time For CA1, CB1, CA2 and CB2 Input Signals.			1.0	µS
T_{CA2}	Delay Time, Clock Negative Transition to CA2 Negative Transition (Read Handshake or Pulse Mode).			1.0	µS
T_{RS1}	Delay Time, Clock Negative Transition to CA2 Positive Transition (Pulse Mode).			1.0	µS
T_{RS2}	Delay Time, CA1 Active Transition to CA2 Positive Transition (Handshake Mode).			2.0	µS
T_{WHS}	Delay Time, Clock Positive Transition to CA2 or CB2 Negative Transition (Write Handshake).			1.0	µS
T_{DC}	Delay Time, Peripheral Data Valid to CB2 Negative Transition.	0		1.5	µS
T_{RS3}	Delay Time, Clock Positive Transition to CA2 or CB2 Positive Transition (Pulse Mode).			1.0	µS
T_{RS4}	Delay Time, CB1 Active Transition to CA2 or CB2 Positive Transition (Handshake Mode).			2.0	µS
T_{IL}	Delay Time, Peripheral Data Valid to CA1 or CB1 Active Transition (Input Latching).	300			nS
T_{SR1}	Delay Time, CB1 Negative Transition to CB2 Data Valid (Internal SR Clock, Shift Out).			300	nS
T_{SR2}	Delay Time, Negative Transition of CB1 Input Clock to CB2 Data Valid (External Clock, Shift Out).			300	nS
T_{SR3}	Delay Time, CB2 Data Valid to Positive Transition of CB1 Clock (Shift In, Internal or External Clock).			300	nS
T_{IPW}	Pulse Width – PB6 Input Pulse	2			µS
T_{ICW}	Pulse Width – CB1 Input Clock	2			µS
I_{PS}	Pulse Spacing – PB6 Input Pulse	2			µS
I_{CS}	Pulse Spacing – CB1 Input Pulse	2			µS

4

Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

Using the ADALAB A/D Converter with Curve Fitter, VIDICHART and Other BASIC Programs

The machine language subroutine listed on the next page allows you to use the ADALAB(tm) A/D converter from a BASIC program. To use this program, enter the ROM Monitor (type CALL -151 in BASIC) and type 320: AD 03 60...etc., copying the hexadecimal numbers from the bottom of the column listing, with a space between each number. Then, return to BASIC (press RESET) and type BSAVE ADOBJ,A$320,L$50. Your BASIC program must begin with LOMEM:24576: D%=0 to ensure that the value of D% is stored at the right place. To read the A/D value, set D% to the slot number of your ADALAB card and CALL 800. On return, D% contains the current A/D value, sampled after the appropriate time delay. The most negative voltage reading is -4095 and the most positive voltage is 4095. Over-range is indicated by -8192 or 8192.

To modify Curve Fitter so that the Control Y command reads the ADALAB A/D converter, type LOAD CURFIT and make the following changes exactly. Then, type SAVE CURFITAD to save this new version.

```
10 HIMEM:38399: LOMEM:24576: D%=0: DIM IN$(50), D(1000), DD(5,2): MX=1000
2900 IF AD=0 THEN SLOT=2: AD=800: PRINT: PRINT CD$"BLOAD ADOBJ,A"AD
2910 D%=SLOT: CALL AD: V0=D%: RETURN
```

To modify VIDICHART(tm) so that the ADC command will access the ADALAB A/D converter, type LOAD VIDICHART and then type the following changes exactly. Then, type SAVE VIDICHARTAD to save this new version. When using the ADC command, the slot number of your ADALAB card must be typed for the channel number (#CHAN). The change in line 948 causes VIDICHART to return to the primary menu when you type Control X during a secondary menu entry.

```
1 HIMEM:35327: LOMEM: 24576: D%=0000: IF PEEK(35411)+PEEK(37232)
    <>128 THEN PRINT CHR$(4)"BLOAD VISIOBJ"
94 CD$ = CHR$(4)
948 IF N<1 OR N>N1-N0+1 THEN PRINT CHR$(7);: GOTO 932
1005 IF OP<1 OR OP>7 GOTO 72
1010 IF AD=0 THEN AD=800: PRINT CD$"BLOAD ADOBJ,A"AD
1015 FOR I=0 TO NS: D%=OP: CALL AD: D%(I,B0)=D%: U=USR(N):
    FOR J=0 TO DLY: NEXT: NEXT: U=USR(H): TEXT: RETURN
```

```
                 0002  ;TITLE ADOBJ
                 0004  ;ADALAB A/D ROUTINE
                 0010  DPER     EQU 6002
                 0020  BASE1    EQU C200
                 0100           ORG 0320
                 0105           OBJ 7000
0320    AD0360   0110  ADCONV   LDA DPER+01
0323    18       0120           CLC
0324    69C0     0130           ADC #$C0
0326    8D3903   0140           STA ADR1+02
0329    8D4603   0150           STA ADR2+02
032C    8D4C03   0160           STA ADR3+02
032F    8D3C03   0170           STA READ+02
0332    8D3F03   0180           STA WAIT+02
0335    A98A     0190           LDA #$8A
0337    8D0CC2   0200  ADR1     STA BASE1+0C
033A    8D00C2   0210  READ     STA BASE1
033D    AD0DC2   0220  WAIT     LDA BASE1+0D
0340    2910     0230           AND #$10
0342    F0F9     0240           BEQ WAIT
0344    AD10C2   0250  ADR2     LDA BASE1+10
0347    8D0360   0260           STA DPER+01
034A    AD20C2   0270  ADR3     LDA BASE1+20
034D    8D0260   0280           STA DPER
0350    2920     0290           AND #$20
0352    D013     0300           BNE PLUS
0354    AD0360   0310  MINUS    LDA DPER+01
0357    49FF     0320           EOR #$FF
0359    8D0360   0330           STA DPER+01
035C    AD0260   0340           LDA DPER
035F    49FF     0350           EOR #$FF
0361    09E0     0360           ORA #$E0
0363    8D0260   0370           STA DPER
0366    60       0380           RTS
0367    AD0260   0390  PLUS     LDA DPER
036A    291F     0400           AND #$1F
036C    8D0260   0410           STA DPER
036F    60       0420           RTS
```

```
]LIST 1000-

1000  GOSUB 80:U =  USR (B0 + 1) +
      USR ( - 3071): POKE XC,0:N =
      78
1005  IF OP < 1 OR OP > 7 GOTO 72
1010  IF AD = 0 THEN AD = 800: PRINT
      CD$"BLOAD ADOBJ,A"AD
1015  FOR I = 0 TO NS:D% = OP: CALL
      AD:D%(I,B0) = D%:U =  USR (N
      ): FOR J = 0 TO DLY: NEXT : NEXT
      :U =  USR (H): TEXT : RETURN
```

```
LABEL  TABLE
DPER     6002
BASE1    C200
ADCONV   0320
ADR1     0337
READ     033A
WAIT     033D
ADR2     0344
ADR3     034A
MINUS    0354
PLUS     0367
```

```
0320- AD 03 60 18 69 C0 8D 39
0328- 03 8D 46 03 8D 4C 03 8D
0330- 3C 03 8D 3F 03 A9 8A 8D
0338- 0C C2 8D 00 C2 AD 0D C2
0340- 29 10 F0 F9 AD 10 C2 8D
0348- 03 60 AD 20 C2 8D 02 60
0350- 29 20 D0 13 AD 03 60 49
0358- FF 8D 03 60 AD 02 60 49
0360- FF 09 E0 8D 02 60 60 AD
0368- 02 60 29 1F 8D 02 60 60
```

Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

MODIFICATIONS OF QUICKI/O TO AVOID INTERFERENCE WITH THE MICROSOFT Z-8Ø SOFTCARD

During initialization of QUICKI/O, the program addresses each interface slot of the Apple computer to determine whether an ADALAB card resides there. Unfortunately, when a Z-8Ø Softcard is used, this turns on the Z-8Ø processor and leaves you in limbo. To avoid this problem, BLOAD QUICKI/O,A$8DØØ and then CALL-151 to enter the monitor. At location $8D7D, you will enter the number of ADALAB cards you have in your system and at $8D7E and thereafter, you will enter the numbers of the slots they occupy, plus $CØ. For example, if you have two ADALAB cards in slots 1 and 4, type the following:

```
8D7D: Ø2 C1 C4
```

Also, you should enter the following patch, which eliminates the search routine:

```
93AD: A9 4C 85 EB 4C E1 93
```

Now, save this modified version of QUICKI/O by typing:

```
BSAVE QUICKI/O,A$8DØØ,L$8FØ
```

Use this version whenever you are using ADALAB together with a Z-8Ø softcard. If you move ADALAB to a different slot, remember to change QUICKI/O as described above.